人物像の歩行動作生成に関する研究

Human Walking Animation

平成 12 年 11 月

筒口 けん

Ken -KENT- Tsutsuguchi

梗概

コンピュータ・グラフィックスによるアニメーションの制作は，様々な分野で応用されている極めて重要な技術である．人物像の動作を計算によって生成することは，コンピュータ・アニメーションの大きな課題であり，現在でも未だ決定的な手法の確立していない分野である．本研究は，計算機内に構築される３次元仮想環境に合致した人物像の歩行動作を生成し，その結果をアニメーションとして表示する手法とその実現方法に関するもので，全６章から構成される．

本研究で用いる手法は，人物像を剛体棒によって構成される階層的な多関節構造体としてモデル化し，連結する剛体棒間の角度を関節角度とみなして，ある時刻における全関節角度をある条件のもとで計算するものである．この人物像の表示は，上記多関節構造体に対応したコンピュータ・グラフィックスによる人物像モデルの各関節に，計算結果である関節角度を入力し，一定の時間間隔ごとに描画することによって行う．この時，時間変数は離散値であり，各関節角度も離散的な時刻ごとに計算する．全関節角度を時間軸方向に連続して記述することにより，人物像の運動を連続した動きとして生成し，表示することになる．

その際，関節角度あるいは角速度の変化を，時間を変数として記述することにより，表示時刻単位で指定しなくても運動を自動的に生成し，表示することが可能となる．しかしながら，人物のように多くの自由度を持つ複雑な多関節構造体の運動を時間を変数とする関数として記述することは困難であり，歩行動作という基本的な運動においてすら決定的な手法は確立していない．特に，階段などを含む複雑な地形に合致する歩行動作を自動的に生成することには大きな困難が伴う．

本研究は，このような問題を解決し，人物像の歩行動作生成を生成するような位置及び関節角度の変化を時間の関数として定式化し，その際の動力学的計算の効率化を行うことによって，地形に合致する人物像の歩行動作を，少ない入力によって，高速に，自然な歩行動作を自動的に生成することを実現したものである．

本研究における第一の特徴は，任意の３次元地形に沿う人物像の歩行動作を計算機によって自動生成する手法の構築である．

計算機による歩行動作の生成は以前から研究されていたが，そのほとんどが，動作範囲が平面上に限定されていたり，周囲の環境とは無関係である歩行のみであった．また，アニメーションを制作する利用者が望む経路に沿った歩行動作を生成しようとすると，入力に多大なコストを要していた．これは，周囲の環境に対する歩行動作の指定方法や，直進歩行や方向転換などの動作生成を，計算効率が良い形で数式化することが困難であったためである．

本研究では，歩行経路を一歩ごとに分割し，一歩の歩幅と高低差を空間的な境界条件とし，また一歩の歩行に要する時間を時間的な境界条件として運動方程式を立てることにより，歩行動作を生成する人体の関節角度の時間変化を，時間の関数として構築することが可能となった．また，歩行動作を構成する方程式を直進歩行動作部分，回転動作部分，そして腕の運動部分に分離し，それぞれの運動を生成した後に互いの運動を融合し，相互作用部分を修正した．さらに，歩行経路の幾何

学的な条件により，関節角度の可動範囲を修正することで，画一的な歩行動作ばかりではなく，誇張表現なども可能となった．

その結果，上下変化や左右への回転を含む複雑な地形に合致する歩行動作を生成することが可能となり，3次元環境に対する歩行経路を設定するだけで，その経路に沿う歩行動作を自動生成することが可能となった．この研究成果により，歩く人物像を容易に環境コンテンツに含むことが可能となり，また，人物像歩行アニメーション制作に対するコストの削減が可能となった．

本研究における第二の特徴は，歩行動作の生成や制御，ならびに生成結果の表示を効率良く行う手法の実現である．これは，個々の歩行者の動作生成や動作制御を効率よく行うことと，多人数の歩行者の動作生成ならびに表示を効率よく行うものである．

一般に，歩行動作生成を含めて，人物像の運動を運動方程式によって生成する場合，境界条件が存在するために方程式の外力項を未知とすることが多く，この外力項を求めるための反復計算に時間がかかってしまい，例えば，歩行の状態を変更するなどの制御が容易に行えない，という問題があった．また，多数の人物像の運動を生成し表示すると，人数に比例して運動生成のコストが増大する，という問題があった．

本研究では，床反力を測定結果に基づいて解析関数を用いて近似し，歩行動作を生成する外力としてその関数を適用することにより，外力項を直接処理することを可能として歩行動作の生成の高速化・安定化を実現した．また，多人数の歩行動作生成を行う際に，各人物像を運動の更新周期の異なる複数のグループに分類して生成することにより，見た目の自然さを損なわずに全体の計算時間を減ずることを可能とした．

この研究成果により，個々の歩行者の歩行動作の計算を高速に，かつ，安定して行うことが可能となり，また，歩行状態の容易な制御を行うことが可能となった．さらに，多人数の運動生成に要する時間を軽減することが可能となり，実時間性を要求するアニメーション・システムへの適用を可能ならしめた．

本研究における第三の特徴は，第一の成果および第二の成果を応用して実際にアニメーション・システムを構築し，様々なコンテンツ制作への適用を実際に行ったことである．即ち，アルゴリズムを構築するのみならず，本研究成果を実現するソフトウエア・システムを構築し，環境コンテンツに対するパス設定のインタフェースや，ネットワークを介した映像の協調制作，2次元画像と3次元CGキャラクタとの合成，あるいは2次元画像で作成したCGキャラクタの生成など，多くの事例に適用した．

その結果，様々なコンテンツに動く人物像を加えることによる視覚的な効果の向上を容易に実現でき，また，人物像歩行動作生成の作業量を軽減することによるコンテンツの生産性の向上を実現した．また，産業展示会やビデオパッケージ制作にも実際に利用されるなど，応用面でも産業界に貢献することができた．

以上の成果により，本研究は，仮想環境に合致する人物像の歩行動作生成手法を確立し，3次元仮想環境に適用する人物像の歩行動作を効率よく生成することを可能とし，それらを実際の映像コンテンツの制作現場に適用することが可能となった．

以下，本論文では，第1章において本研究の概要と目的を述べ，第2章において関連する手法を述べ，本研究の位置付けについて論ずる．第3章では，本研究における基本的な用語，歩行に関する基本的な事項について述べ，3次元環境内において歩行パスの設定のみで歩行動作を生成する手法に付いて述べる．第4章では，歩行動作の効率的な制御および生成について述べる．まず，歩行動作を生成する外力としての床反力の適用について説明し，次いで多数の歩行者を生成・表示する際のコストの削減について述べる．第5章では，本研究を実装するシステムの構成と人物像モデルに付いて説明し，実際にシステムを構築して様々なアプリケーションに適用した例を報告する．その後，第6章で本研究の有効性について検討を行い，本論文のまとめを行う．

本研究は，アニメーション制作，Webコンテンツ制作などの映像コンテンツはもちろんのこと，都市景観コンテンツ制作や建築設計など，ディジタル・コンテンツの制作に関する様々な分野に適用可能である．本研究で一貫して追求しているのは，いかに少ない人力，少ない計算時間で，最大限の効果を上げるような歩行動作生成システムを提供できるか，という制作者側に立った成果であり，実際の制作現場において，その責を果たしている．本研究は，このようなディジタル・コンテンツ生産の支援を通じて産業界ならびに文化生活に寄与するものである．

目次

1 序論 **1**

2 背景 **7**

2.1 はじめに 7
2.2 アニメーション手法 7
2.3 人物像の運動生成 9
2.4 歩行動作生成手法 13
2.5 本研究の位置付け 15

3 地形に沿う人物像歩行動作の生成 **19**

3.1 はじめに 19
3.2 人体の構造 19
3.3 歩行に関する準備 21
3.3.1 歩行時の持続時間 22
3.3.2 歩行の幾何的境界条件 23
3.3.3 歩行時のパラメータ 24
3.4 直線歩行に対する人物像歩行動作の生成 25
3.4.1 動力学変数および運動方程式 27
3.4.2 歩行動作生成の手順 27
3.4.3 歩行パスの分割 29
3.4.4 境界条件の計算 31
3.4.5 歩行動作の生成 33
3.4.6 表示 38
3.4.7 生成結果例 39
3.4.8 検討 40
3.5 腕の動作の生成 41
3.5.1 腕の運動に対する運動方程式 43
3.5.2 肘角度の計算 46

3.6 部分運動への分離と再構成 . . . 47
3.6.1 パスの分解 . . . 48
3.6.2 動力学変数 . . . 50
3.6.3 回転運動 . . . 51
3.6.4 ステップの連結 . . . 53
3.6.5 誇張表現 . . . 53
3.6.6 生成例 . . . 56
3.6.7 検討 . . . 61
3.7 本章のまとめ . . . 62

4 歩行動作の効率的な生成および制御 65
4.1 はじめに . . . 65
4.2 床反力を用いた歩行動作の生成 . . . 65
4.2.1 床反力の分析 . . . 66
4.2.2 床反力を用いた歩行動作生成 . . . 70
4.2.3 生成結果例と検討 . . . 76
4.3 更新周期制御による多人数の歩行動作の生成 . . . 77
4.3.1 一歩行者の動作生成 . . . 80
4.3.2 群衆としての歩行者 . . . 81
4.3.3 運動の削減 . . . 82
4.3.4 生成例と検討 . . . 84
4.4 本章のまとめ . . . 88

5 アニメーション・システムの実装および応用例 91
5.1 はじめに . . . 91
5.2 アニメーションシステム WorldWideWalk . . . 91
5.2.1 システムの構成 . . . 91
5.2.2 WorldWideWalk における処理の流れ . . . 94
5.2.3 人物像モデル . . . 95
5.2.4 3次元室内シーンへの適用例 . . . 96
5.3 3次元環境でのパスの自動設定 . . . 97
5.3.1 歩行経路の自動設定 . . . 100
5.3.2 障壁における方向転換 . . . 101
5.3.3 生成例と検討 . . . 102
5.4 2次元背景画像との合成 . . . 106
5.4.1 歩行領域の設定 . . . 106

5.4.2 生成例と検討 . . . 107
5.5 ネットワークを介した生成例 . . . 108
5.5.1 プロセスモデル . . . 113
5.5.2 生成例 . . . 114
5.6 画像を用いた人物像モデルを用いた生成 . . . 115
5.6.1 人物像モデルの生成 . . . 116
5.6.2 アニメーション生成例 . . . 116
5.7 本章のまとめ . . . 119

6 結論 123

謝辞 127

参考文献 128

A C/C++ インプリメンテーション 139
A.1 人体パーツのヘッダファイル . . . 139
A.2 人体の位置および関節角度データのヘッダファイル . . . 142
A.3 腰部を基準にした人物像の描画を行うコード . . . 144

B Java/Java3D インプリメンテーション 147
B.1 人体パーツを定義するクラス . . . 147
B.2 人体の位置および関節角度データを定義するクラス . . . 149
B.2.1 フレーム毎のデータ . . . 149
B.2.2 1歩毎のデータ . . . 149
B.2.3 歩行開始から歩行終了までのデータ . . . 150
B.3 腰部を基準にした人物像の描画を行う疑似コード . . . 152

本研究に関する業績 155

表一覧

3.1 人物像モデルを構成するデータ（数字は自由度） 21
3.2 境界条件例 55
3.3 1歩に要する平均計算時間 (単位: 秒); (a) 直線歩行, (b) 平地曲線歩行. 61

図一覧

1.1 歩行動作生成システムの概念. 3

2.1 コンピュータ・アニメーションの一例. 8
2.2 キーフレーム法の原理. 9
2.3 運動学による関節構造体の関節角度の指定. 10
2.4 動力学によるアニメーション生成例; 球の自然落下. 12
2.5 本研究における人物像の動き生成の概念. 16

3.1 多関節構造体としての人物像モデルの構造. 20
3.2 動作生成の基本的な考え方; 動力学でおおまかな運動を生成し, 運動学で調整を行う. 22
3.3 歩行に関する持続時間. 23
3.4 歩行時の境界値. 24
3.5 高低差ある地形への適用; (a) 平地, (b) 複雑な地形. 25
3.6 人物像モデルと動力学変数 (a) 人物像モデル, (b) 座標系と動力学変数. 26
3.7 動力学変数. 28
3.8 地形への適応例. 29
3.9 歩行パスの近似. 30
3.10 エリア分割とステップ分割. 31
3.11 直進歩行時の境界条件. 33
3.12 遊脚足首の軌道. 36
3.13 立脚足部角度の計算; (a) 踵着地, (b) 爪先着地, (c) 他の足の踵着地, (d) 両足支持. . 37
3.14 起動後の GUI 画面. 38
3.15 3 次元地形例; (a) 上部図, (b) 俯瞰図. 39
3.16 パスの設定と表示; (a) 上部図, (b) 俯瞰図. 40
3.17 パスに沿った歩行動作の生成例. 41
3.18 歩行シーケンス生成例. 42
3.19 腕の変数; (a) 上腕・下腕の回旋と脇の角度, (b) 肩, 前後への角度, 肘角度. 43
3.20 腕のモデル; (a) 人体モデル, (b) 前後への運動, (c) 肘の運動, (d) 腕の広がり. . . . 44

3.21 腕全体の運動モデル 45
3.22 腕の幾何的近似モデル 46
3.23 境界条件および肘の運動 46
3.24 パスの分解. 48
3.25 回転パス. 49
3.26 パス設定及びステップへの分割の例. 49
3.27 動力学変数. 50
3.28 境界条件. 51
3.29 回転のモデル. 52
3.30 回転の動力学モデル. 53
3.31 ステップの歩行モード. 54
3.32 境界条件の修正例; (a) 上下方向, (b) 左右方向. 55
3.33 回転に対する体軸の変化例 57
3.34 人物像モデル設定の GUI. 57
3.35 曲線歩行例; (a) パスの設定, (b) パスのステップへの分割, (c) 動作生成. 58
3.36 誇張表現の因子設定例. 58
3.37 体軸の傾きを変更して生成した例 (1). 59
3.38 体軸の傾きを変更して生成した例 (2). 60
3.39 階段を下降しながら方向転換を行うアニメーションの生成例. 63

4.1 1歩に対する典型的な床反力. 67
4.2 持続時間と床反力. 68
4.3 鉛直方向の床反力測定結果と近似曲線. 68
4.4 床反力近似関数. 70
4.5 脚部の動力学変数. 71
4.6 直進歩行に対する境界条件. 72
4.7 立脚の補間（再掲）; (a) 踵着地, (b) 腰が足首の真上に来た瞬間, (c) 他の足の踵着地, (d) 爪先離地. 73
4.8 人物像重心の物理状態; (a) 加速度, (b) 速度の平均移動速度に対する割合, (c) 位置. 74
4.9 回転運動における一般化力関数の近似. 75
4.10 腕の運動における一般化力関数の近似. 75
4.11 生成結果例; (a) 階段の下降, (b) 階段の上昇, (c) 平地での曲線歩行. 76
4.12 生成結果例; 上段から下段へ, 左から右へ進む. 78
4.13 京都市祇園の CG 画像: (a) “ディジタルシティ京都” からの画像, (b) 歩行者を加えたもの. 79
4.14 WALKER モジュール. 81

4.15 歩行者の視点から見たシーン. 82
4.16 本手法の概念; (a) 視界, (b) 表示ウィンドウ. 83
4.17 グループの概念. 83
4.18 提案手法のフローチャート. 84
4.19 実験したグループ例. 85
4.20 200 人の歩後者に体する計算時間 (100 フレーム); (a) 1 グループ, (b) 各種のグループパターン. 86
4.21 アニメーションシーン (負荷レベル =1000), t は 100 フレームごとの計算時間; (a) 歩行者なし, (b) 10 人, (c)100 人, (d) 300 人. 87
4.22 人気のある場所の表示例. 88

5.1 システムの階層構造. 92
5.2 歩行動作生成システムのブロック構成図. 93
5.3 歩行動作生成の処理の流れ. 94
5.4 人物像モデル例. 95
5.5 Bezier 曲面を用いた人物像モデル例. 数字は Bezier パッチ数を表す. 95
5.6 Bezier 曲面. 96
5.7 *WorldWideWalk* による生成例; (a) パス設定, (b) パラメータ計算, (c) 生成結果. . 97
5.8 室内シーンでの生成例; (a) パス設定, (b) ステップへの分割, (c) 階段下降, (d) 室内歩行, (e) 複数の歩行者生成. 98
5.9 ビデオ作品「梅美月」での歩行シーン. 99
5.10 障壁オブジェクトの概念. 101
5.11 壁面での反射モデル. 102
5.12 鏡面反射モデル; (a) パス生成; ハイライト部分が初期位置と方向を指定する部分, (b) ステップへの分割. 103
5.13 ランダム反射モデル; (a) パス生成; ハイライト部分が初期位置と方向を指定する部分, (b) ステップへの分割. 104
5.14 複数歩行者の生成例; (a) 鳥瞰図, (b) 歩行者の視点. 105
5.15 一点透視図によって撮影された画像に対する，仮想的な奥行き情報の付与. 107
5.16 二点透視図によって撮影された画像に対する，仮想的な奥行き情報の付与. 108
5.17 歩行領域とカメラパラメータの設定例; (a) 歩行領域の設定, (b) サイズと縦横比の設定, (c) 合成例. 109
5.18 2次元画像との合成例 (1); (a) 背景画像, (b) スケルトンモデルを用いた合成例, (c) ポリゴンモデルを用いた合成例. 110
5.19 2次元画像との合成例 (2); (a) 背景 CG 画像, (b) 合成例. 111
5.20 VMODELER による出力画像との合成例 (1). 112

5.21 VMODELER による出力画像との合成例 (2). 112
5.22 ネットワーク・モデル. 113
5.23 プロセス・モデル. 114
5.24 ネットワークを介したアニメーション生成例. 115
5.25 画像とスケルトン情報，パーツ情報との対応. 117
5.26 画像を用いた人物像モデルによる歩行動作の生成例; (a) 原画像, (b) スケルトン情報の付与, (c) パーツ情報の付与, (d) スケルトンに対する歩行動作生成, (e) 生成例. 118
5.27 画像を用いた人物像モデルによる多人数の歩行動作生成例. 119
5.28 CG キャラクタ画像; (a) 正面図, (b) 背面図. 120
5.29 CG キャラクタを用いた歩行動作生成例. 121
5.30 「21 世紀 夢の技術展」に展示されたシステムで制作されたアニメーション例. . . . 122

第 1 章

序論

絵画，彫刻，音楽，文学など，古来，あらゆる文化活動において人間は人間自身を様々な表現の対象としてきた．フィルムやビデオのような時間軸方向に記録が可能な記録媒体が発明されてからも，人間は常に最も重要な表現対象であり続けた．ディジタル技術が進歩し，様々なコンテンツが電子的に作成されるようになった今日においても，人間にとって人間自身は非常に重要で魅力的な創作および表現対象であり，未来においてどのように技術が進歩してもそうあり続けるであろう．

そのような中で，計算機を用いて様々な画像を生成し描画するコンピュータ・グラフィックス（CG）は，記録，複製や編集などが極めて容易であることから，ディジタル・コンテンツの生産において重要な役割を果たしている．近年はコンピュータ・グラフィックス技術の産業上の応用が特に増大しており，映画や TV などにおける特殊効果や仮想的なキャラクタ制作への応用，TV ゲームやアーケードゲーム，様々な科学計算やシミュレーション結果の表示，ネットワーク上でのコンテンツ制作など，枚挙にいとまがないほどである．

コンピュータ・グラフィックスにおいても，人物像をはじめとするキャラクタの動作生成は重要なテーマである．近年のハードウエア技術の急速な進歩ならびにコンピュータ・グラフィックス技術の発達により，様々な用途向けの高品質な映像シーンが生成できるようになったが，複雑な動物体である人物像の自然な動作を計算機によって生成することは，多くの点が未解決であり，未だ決定的な手法の確立していない分野である．

人物像の生成に関しては，以下の３つの重要な課題があり，それぞれ，様々な研究例が報告されている：

（a） 人体の形状の生成（表面テクスチャ等を含む），
（b） 人間の運動の生成，あるいは制御，
（c） 仮想世界における周囲の状況あるいは環境との種々の相互作用．

セル画によるアニメーションに代表される古典的手法では（a），（b），（c）の明確な区別はなく，全てアニメータの能力に依存していた．近年，３次元入力装置等のハードウエアの進歩により，アニメータに依存しない人体形状自動生成手法が開発されるにつれて骨格モデルなどによる運

動の生成手法（（b）の要素）と（a）の要素が次第に分離されてきている．

（a）の「人体の形状の生成」は，3次元形状データ，人体表面のカラー，筋肉，頭髪，衣服のモデリングおよびレンダリングが含まれる．即ち，「写実的な人物像」あるいは「デフォルメされた個性的なキャラクタ」などを生成するためのものであり，コンピュータ・グラフィックスおよびコンピュータ・アニメーションにおける重要な研究課題である．

（b）の「人間の運動の生成，あるいは制御」は，動画という表現方法が登場して以来，非常に重要な技術となっている．人間に限らず，様々なキャラクタの自然でスムーズな動き，あるいは魅力ある動きを生成することの重要性は冒頭でも述べた通りである．

（c）の環境との相互作用は，コンピュータ・グラフィックスにおいて比較的新しい概念である．従来の人物像の運動を扱ったコンピュータ・アニメーションにおいては，周囲の状況という要因は考慮されていないか，あるいは相互作用の対象が限定されていたものがほとんどである．その理由として，コンピュータ・グラフィックスにおける仮想環境は現実世界と異なり，ある程度単純化あるいは理想化されたものであり，限定された各種条件を数値化したシミュレーションなどに適している反面，現実に起こり得ないことも生ずることがあるため（例えば壁を突き抜けて歩いて行ってしまう，など），環境やオブジェクトの物理特性を詳細に，そして明確に記述する必要があり，一般にそれらの間の相互作用を記述することは困難なことが挙げられる．しかしながら，動作生成を考慮していくにあたり，仮想世界における環境との関わりは極めて重要な要素であり，考慮しなければならない問題である．

本論文では，計算機による人物像の歩行動作の生成に関する研究について述べる．本研究は，（b）および（c）の課題を実現しようとするものであり，与えられた3次元仮想空間内に設定された任意のパスに沿った人物像の歩行動作を自動的に生成する手法を実現しようとするものである．

歩行動作は人物の重要な基本動作のひとつであり，映像制作，キャラクタ・アニメーション作成，各種シミュレーション，CAD 等への適用，人間工学的な解析など様々な応用面で極めて需要が高い．歩行動作を自動的に生成することは，リアリティあふれる映像表現を可能とするために極めて重要である．その一方で，ウォークスルー・アニメーションに代表される環境コンテンツの表現においては，人物像の重要性が指摘されながらも人物像が含まれていないことが多く，また含まれていても静止しているか環境とは関係のない動きをしていることが多い．

本研究では，坂道や階段昇降などを含む，現実世界と同じような環境への適応を実現する歩行動作生成手法の構築を行い，人物像の動きだけの生成や，環境モデルの生成だけではなく，それらを融合した映像コンテンツを生成することを目指している．図 1.1 は本研究の基本コンセプトを示している．環境に適応できる歩行動作を自動的に生成することにより，建築物等の CG 環境に歩行する人物像を加えて現実感を出したり，歩く人物像をリファレンスとして様々な設計に適用したり，実際に映像コンテンツを制作するなど，幅広い応用面が期待できる．

その際，

1. できるだけ少ない作業量によりアニメーションを生成すること，
2. できるだけ少ない計算量によりアニメーションを生成すること，

の2点を実現することにより，アニメーション制作を行うアニメータの負担を軽減し，また制作にかかるコストを軽減することになる．それには，環境コンテンツに合致した人物像の歩行動作を，なるべく少ない入力パラメータで，かつ，なるべく計算効率の良い形で自動生成するアルゴリズムの構築が必要である．

本研究における運動の生成は，運動の状態を数値化して計算機によって計算し，計算結果をディスプレイ上に表示することを前提としている．人物像を互いに連結する剛体棒によって構成される階層的な多関節構造体としてモデル化し，連結する剛体棒間の角度を関節角度とみなして，ある時刻における全関節角度をある条件のもとで計算することにより，その時刻の人物像モデルの姿勢を決定する．その姿勢を連続して一定の時間間隔ごとに表示することにより，人物像の運動を表現するアニメーションを生成する．この時，時間変数は離散値となり，各関節角度も離散的な時刻ごとに計算することになる．

その際，全関節角度に対する運動方程式を時間を変数として記述し，境界条件を与えて実際に数値計算を行い，各時刻における人物像の姿勢を，時間軸軸方向に連続して表示することにより，コンピュータ・アニメーションとしての人物像の運動を自動的に生成することが可能となる．

しかしながら，人物のように多くの自由度を持つ複雑な多関節構造体の運動を計算可能な形で記述することは困難であり，歩行動作という基本的な運動においてすら決定的な手法は確立していない．特に，階段などを含む複雑な地形変化に合致する歩行動作を自動的に生成することには，人物像の関節角度の記述のみならず周囲の地形やそれに対する対応手段も記述する必要があるため，大きな困難が伴う．

本研究の目的は，このような問題を解決するために，地形に合致する自然な歩行動作を，少ない入力によって高速に計算機で自動的に生成できるように定式化し，実際に様々な環境コンテンツのもとでの歩行動作の生成を実現することである．

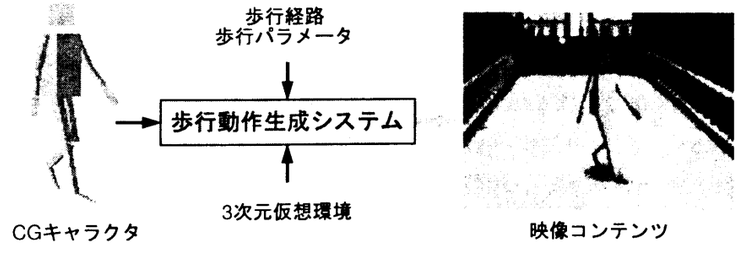


図 1.1: 歩行動作生成システムの概念．

本研究における第一の特徴は，任意の３次元地形に沿う人物像の歩行動作を計算機によって自動生成する手法の構築による，作業量の削減である．

計算機による歩行動作の生成は以前から研究されていたが，そのほとんどが，動作範囲が平面上に限定されていたり，周囲の環境とは無関係である歩行のみであった．また，アニメーションの制作者が，望む経路に沿った歩行動作を生成しようとすると，入力に多大なコストを要していた．これは，周囲の環境に対する歩行動作の指定方法や，直進歩行や方向転換などの動作生成を，計算効率が良い形で数式化することが困難であったためである．近年はモーション・キャプチャー装置を用いて実際に人間の動き情報を採取し，CG キャラクタに適用する事例が増えているが，装置から受ける制約が大きかったり，採取した動きを他の動きに適用できないなどの問題がある．

本研究では，歩行経路を一歩ごとに分割し，一歩の歩幅と高低差を空間的な境界条件とし，また一歩に要する時間を時間的な境界条件として運動方程式を立てることにより，歩行経路に沿った歩行動作を行う人物像の関節角度の時間変化を，時間の関数として構築する手法を提案する．高低差ある地形への対応は，既知の地形データに沿って設定した移動経路に合致するように，経路を同じ高さを有するまとまりに分割する “エリア分割” と，さらにそれを一歩ごとに分割する “ステップ分割” により解決する．歩行動作の生成は，Bruderlin らによる平地の歩行動作生成手法を拡張した，動力学的手法と運動学的手法を融合させた手法 [19] によって行う．

また，本研究では，直進歩行動作のみならず，曲線経路にも対応するために，歩行動作を構成する方程式を直進歩行動作部分，回転動作部分，そして腕の運動部分に分離し，それぞれの運動を生成した後に互いの運動を融合し，相互作用部分を修正する手法を提案する．併せて，３次元歩行経路も直進成分と回転成分とに分解し，歩行経路の直進成分に対して直進歩行動作生成部分を，歩行経路の回転成分に対して回転運動生成部分を適用することで，任意の３次元歩行経路に合致した歩行動作の生成を行う．さらに，歩行経路の幾何的な形状から得られる条件により，関節角度の可動範囲を修正することで，画一的な歩行動作ばかりではなく，歩行パスに応じた誇張表現も可能とする．

この研究成果により，上下変化や左右への回転を含む複雑な地形に合致する歩行動作を生成することが可能となり，３次元環境に対する歩行経路を設定するだけで，その経路に沿う歩行動作の自動生成を実現した．その結果，歩く人物像を容易に環境コンテンツに含むことが可能となり，人物像歩行アニメーション制作に対する作業コストの削減を実現した．

本研究における第二の特徴は，歩行動作の生成や制御に対する計算を効率良く行う手法の実現である．即ち，個々の歩行者の動作生成や直接的な制御を少ない入力，少ない制御パラメータで高速に実現する手法と，多人数の歩行者の動作生成ならびに表示を少ない計算コストで実現する手法とを提案する．

一般に，歩行動作生成を含めて，人物像の運動を運動方程式によって生成する場合，境界条件に合致するように外力を加える必要があるため，この外力項を未知として反復計算を行なうこと，即ち，いわゆる逆問題を数値的に解くことを行なわなければならず，計算に時間がかかっていた．また，上記の理由から外力を直接制御することが容易ではないため，歩行状態を変更する場合，関節角度の可動範囲を変更することだけで対応するなどの制限があった．さらに，多数の人物像の運動を生成し表示すると，人数に比例して描画コストが増大する，という問題があった．このため，複数の動作生成方法を用意しておき，それらを切替える手法が提案されているが，動作生成方法のアルゴリズムを変更せずに計算時間を減ずる手法は見られず，個人や集団の歩行動作の制御や表示を，実時間に近いレベルで実現する手法が望まれていた．

本研究では，人間が歩行する際の床反力を実際に測定し，測定結果に基づいて床反力をフーリエ級数で近似して様々な地形に応じた床反力関数を得，歩行動作を生成する運動方程式の外力項としてその関数を適用する手法を提案する．

さらに，本研究では，多人数の歩行動作を生成する際に，各人物像を運動の更新周期の異なる複数のグループに分類して生成および表示を行う手法を提案する．即ち，前述の床反力を用いる制御手法を採用することによって個々の歩行動作生成を効率化し，さらに，各歩行者を表す人物像のグラフィックス・オブジェクトを，ディスプレイに表示する際の仮想的な視点からの距離に応じて複数のグループに分け，視点から近いオブジェクトは毎フレームごとに運動の更新を行い，視点から遠いオブジェクトは数フレームごとに運動の更新を行う手法を用いる．

この研究成果により，個々の歩行者の歩行動作の計算を行う際に，動力学計算時に生じ得る数値的な不安定性を解消し，また，歩行動作を生成する運動方程式に現れる外力の項を直接制御することにより，歩行動作計算を高速化・安定化ならしめると同時に，歩行動作の直接的な制御が可能となった．さらに，多人数の運動生成ならびに表示を行う際に，各歩行者の生成ならびに制御を効率的に行と同時に，見た目の自然さを損なわずに全体の計算時間を削減することを実現し，実時間性を要求するアニメーション・システムへの適用が可能となった．

本研究における第三の特徴は，第一の成果および第二の成果を取り入れて実際にアニメーション・システムを構築し，様々なコンテンツ制作への適用を実現したことである．

即ち，アルゴリズムを構築するのみならず，本研究成果を実現するソフトウエア・システムを構築し，コンピュータ・グラフィックスで作成された環境コンテンツに合致するような人物像歩行動作の生成，制御，表示，確認，保存などを可能とする制作環境の構築を実現した．例えば，環境コンテンツに対するパス設定のインタフェースを構築したり，ネットワークを介した映像の協調制作，２次元背景画像と３次元キャラクタとの融合，２次元画像を用いた人物像キャラクタによる歩行動作の生成など，種々の機能を実装することで，様々なコンテンツの制作事例に適用した．

その結果，様々なコンテンツに動く人物像を加えることによる視覚的な効果の向上を容易に実現することが可能となり，人物像歩行動作生成の作業量を軽減することによるコンテンツの生産性の向上を実現した．これらのアニメーション・システムは，ビデオパッケージの作成や産業展示会な

どに実際に利用されており，その有効性を示すことができた．

以上の成果により，本研究は，仮想環境に合致する人物像の歩行動作生成手法を確立し，3 次元仮想環境に適用する人物像の歩行動作を効率よく生成し，それらを実際の映像コンテンツの制作現場に適用することが可能となった．

以下，本論文では，第 2 章において関連する手法を述べ，本研究の位置付けについて論ずる．

第 3 章では，本研究における基本的な用語，歩行に関する基本的な事項について述べ，3 次元環境内において歩行パスの設定のみで歩行動作を生成する手法に付いて述べる．まず，高低差ある地形における直進歩行動作の生成法について述べ，腕の運動の生成についてその定式化を行う．その後，左右の方向転換を可能とする歩行動作生成法について述べる．

第 4 章では，歩行動作の効率的な制御および生成について述べる．まず，歩行動作を生成する外力としての床反力の適用について説明し，次いで多数の歩行者を生成・表示する際のコストの削減について述べる．

第 5 章では，本研究を実装するシステムの構成と人物像モデルに付いて説明し，実際にシステムを構築して様々なアプリケーションに適用した例を報告する．適用例には，2 次元画像と 3 次元人物像モデルとの合成手法やネットワークを介したアニメーションの協調生成，人物写真画像を用いた歩行者モデルの生成法などを含む．

その後，第 6 章で本論文の有効性について検討を行い，本論文のまとめを行う．

なお，付録として，本研究をアニメーション・システムとして実装する際の，C／C++ 言語および Java 言語でのヘッダファイルまたはクラスファイルと，ある姿勢の人物像を表示する疑似コードを示す．

本研究は，アニメーション制作，Web コンテンツ制作などの映像コンテンツはもちろんのこと，都市景観コンテンツ制作や建築設計など，ディジタル・コンテンツの制作に関する様々な分野に適用可能である．本研究で一貫して追求しているのは，いかに少ない入力，少ない計算時間で，最大限の効果を上げるような歩行動作生成システムを提供できるか，という制作者側に立った成果であり，実際の制作現場において，その責を果たしている．本研究は，このようなディジタル・コンテンツ生産の支援を通じて産業界ならびに文化生活に寄与するものである．

第 2 章

背景

2.1 はじめに

本章では人物像動作生成の関連手法とその背景について詳しく述べる．まず，アニメーション技術について述べた後，人物像の運動の生成手法について述べる．次いで歩行動作に関する関連手法について述べ，本論文で述べる手法との関連性について議論する．

本論文では，途中段階の生成手法に関わらず，ある連続する運動の状態を離散的な時間間隔ごとに 2 次元画像として生成・描画し，表示する技術をアニメーションと総称する．また，描画対象を描画すべく，色や形状，骨格構造，運動状態など，何らかの特徴を抽出して模した抽象的なものをモデルと呼び，描画対象をモデル化することをモデリングと呼ぶ．

また，本論文において，運動の生成とは，描画の対象となる物体（人物や動物を模したものであったり，自然現象であったり，あるいは単なる幾何的図形であったりする）の，ある時刻における位置や速度などの物理量を決定することである．これは，一般的なアニメーション手法では必ずしも自然法則に従う必要は無く，例えば手描きアニメーションの場合は，アニメータ即ち描き手の感性に従って運動状態が生成され，誇張表現などが行われることもある．

さらに，本論文において，画像の生成あるいは描画とは，生成されたシーンを 2 次元画像座標値に変換し，2 次元の色情報あるいは形状の情報としてフィルムやセルに描き込んだり，計算機のメモリ上に色情報や奥行情報として記憶することを言い，特に計算機の場合はレンダリングと称する．表示とは，これら記憶あるいは記録された 2 次元画像を，実際にディスプレイ上で可視化したり，フィルムであれば光を当てることでスクリーン上に投影し，人間が見えるようにすることを言う．

2.2 アニメーション手法

標準的なアニメーションでは，1 秒間を十数から 30 の，フレームと呼ばれる描画単位に等分割し，各フレームごとにその時刻に対応した描画対象の色や位置，形状を表示する．これは，人間の目が 1 秒間あたりのフレーム数が 20 程度になると連続的な変化と区別することができなくなるた

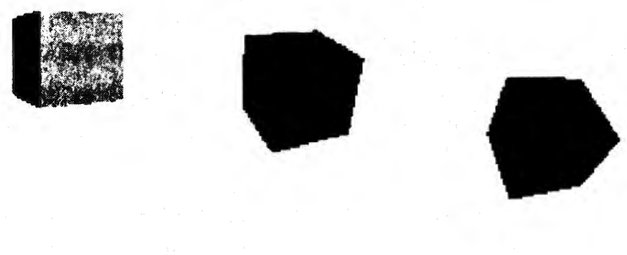


図 2.1: コンピュータ・アニメーションの一例.

めである．このもっとも単純な例がいわゆる「パラパラ漫画」である．従って，フレーム間での運動状態の変化が小さければ，スムーズな動きを得ることができる．しかし，動きの連続性や変化のスムーズさが損なわれたりすると，不自然な印象を与えてしまう．

現在のシステムでは，アニメーションの制作過程や描画対象の次元に拘らず，最終的に 2 次元の画面上に表示されることを前提としているものがほとんどであるが，モデリングの過程で分類すると，描画対象モデルの有する空間情報が 2 次元であるものを 2 次元アニメーションと呼び，3 次元であるものを 3 次元アニメーションと呼ぶ．セル画によるアニメーションに代表される古典的手法では，描画対象のデザインや運動の生成ともに 2 次元アニメーションである．そこでは，計算機を用いた自動生成手法も採り入れられつつあるが，依然として手描きが主体であり，描画対象の色，形状，運動いずれのモデル化も，アニメータの能力に依存するところが大きい．

近年，モデルの再利用性や様々な視点から見ることの利点を活かして，3 次元アニメーションによる制作が増加している．特に，コンピュータ・ハードウエア技術，並びにコンピュータ・グラフィックス技術の発達により，計算機を用いて，様々な用途向けの高品質な映像シーンが生成できるようになってきており，全編 3 次元アニメーションで制作された映画が登場したり，自然現象や植物の動き，爆発シーンなどの特殊効果，過去の生物や想像上の生物，各種乗物など，非常に多くのシーンの制作に応用されている．3 次元コンピュータ・アニメーション全般に関しては，Foley ら [30] や O'rourke [71] の本に詳しい．

図 2.1に 3 次元コンピュータ・アニメーションの例を示す．この図では，左から右へ，運動する立方体の動きを生成している．このような動きを生成するための，最も伝統的な手法は**キーフレーム法**（中割法）と呼ばれる手法である．これは，あらかじめ初期の状態と目的となる状態の画像を指定しておき，何らかの法則でそれらを補間する手法である．即ち，伝統的なセルアニメーションの中割の描画をコンピュータが代用する．図 2.2に最も単純なキーフレーム法の原理を示す．図 2.2の「Keyframe 1」が初期位置，「Keyframe 2」が目的位置であり，「interpolation」は，その中間位置を線形に補間している．

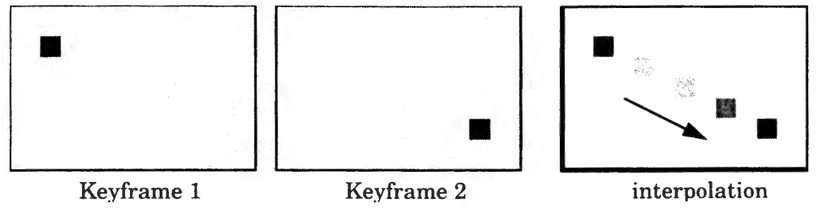


図 2.2: キーフレーム法の原理.

さて，人物像を対象とした 3 次元アニメーションには，表面テクスチャ等を含む人体の色，形状の生成に関するものや，動き，運動の生成あるいは制御に関するものがある．人体の色や形状の生成に関する手法には，3 次元形状データ，人体表面のカラー，筋肉，頭髪，衣服の生成等を含めることができる．即ち，「見るからに人間そっくり」に生成したり，より個性あるキャラクタを制作するためのものであり，魅力的な人物像を生成する上で重要なファクターである．顔に関しては，レーザーを用いたり，あるいはコンピュータ・ビジョン技術を用いて 3 次元の顔モデルを入力する装置も普及してきている．また，実写顔画像や顔の筋肉の動きデータを用いて，よりリアルな顔モデルを生成して表情をつける手法が Guenter ら [41] や Pighin ら [79] によって報告されている．

一方，人物像の動きを生成する手法では，例えば [7] や [8] にまとめられているが，人物像を点で接続された剛体棒で近似し，各剛体棒間の相対的な位置関係を記述するものがほとんどである．後で詳しく説明するが，一定の長さと一定の質量を有する剛体棒が関節を表す点で接続されて階層構造をなすものを多関節構造体と称する．これ以降，本論文では人物像の運動の生成というときは，3 次元空間における多関節構造体の運動を表すものとし，本研究で扱う歩行動作生成手法もこの範疇に属する．

2.3 人物像の運動生成

コンピュータ・アニメーションによる多関節構造体としての人物像動作の生成手法は，主として運動学的手法と動力学的手法に大別される．

運動学（kinematics）とは，個々のオブジェクトの位置的な相互関係，即ち，「あるオブジェクトの位置を変化させた場合，他のオブジェクトの位置はどう変化するか」を記述する手法である．人物像のような多関節構造体の場合，各フレームにおける身体の各部位の位置や，部位間の関節角度などを指定することになり，それらの位置や関節角度の変化がどのような法則に従っているかは問わない．極端に言えばどのような動きでも作成することが可能である．

図 2.3は関節構造を持つオブジェクトの，運動学による姿勢決定の例を示す．上部の図のように，構造の根本（ルート）に近い方から末端（エンド）の方へ，回転角を順次与えていくような方法を順運動学（forward kinematics）と言う．この方法は，全ての関節角度を与えれば，結果が必ず一

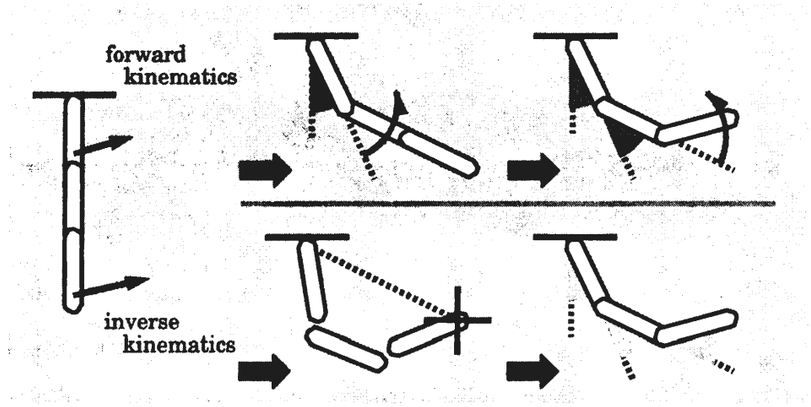


図 2.3: 運動学による関節構造体の関節角度の指定.

意的に決まる．この手法を用いてアニメーションを生成する場合は，各時刻における全関節角度を計算することになる．この手法の欠点は，全ての関節角度を与えなければならない点と，エンド部を望ましい位置に到達させることが難しいことが挙げられる．

一方，下の図のように，先にエンドの目的地を決めておいて，それに合うように各関節の角度を決めるような方法を逆運動学（inverse kinematics）と言い，人物像の動きの生成に関しては，逆運動学を用いた手法が多い．この手法は，単に動かすだけでなく，ものを掴むなどの目的を持った（goal-oriented）動作を生成するのに適しており，実際の CG ソフトでも多く採用されている．しかし，例えばエンドの位置がグローバルな座標系でどの（絶対）位置にあるか，を常に把握しなければならないことと，関節の数が増えると各関節角度の値が一意的に決まらない，という問題がある．このため，さまざまな拘束条件を加えて，なるべく各関節の角度が一意的に決まるように工夫がなされている．

運動学的手法は，自然な運動や環境シーンへの自動的な適応などには不向きであるが，計算時間や対話的な動作編集に有利であるため，広く使用されている．しかしながら，運動学的手法は，動きの質が制作者の技量に依存し，作業量が大きくなるという問題がある．このため，あらかじめ各部位の運動を作成しておき，それを構成して全身の運動を構成する手法が早くから提案されている．例えば，Zeltzer は人物像の動作を構成要素である複数の LMP（local motor program）に分解し，各 LMP を運動学的に表現することで歩行動作を生成している [116][117]．Thalmann，Boulic らは生体工学的データを基に逆運動学的手法で目的の動作を生成し [93][94]，福井らは逆運動学の行列を近似的に計算して滑らかな運動を表現しようと試み [33][34]，Calvert らは運動の構成単位を対話的手法で組み合わせて人物像の動き生成システムを開発している [24][25]．また，Boulic らは運動の軌道曲線を修正することによってスムーズな運動を実現しようとし [15]，Witkin ら [114] や Bruderlin ら [22][23] は，異なる全身運動の間を滑らかに連結する手法を提案している．

一般化された動作の生成のみならず，個性ある動作や多岐にわたる動作を生成する手法の研究も行なわれており，Trappl らによってまとめられている [96]．Perlin はライブラリ化された動きにより，実時間で個性ある動きを表現しようとし [76]，また，特定の動作をライブラリ化し，スクリプト言語による動作記述を試みる研究も Perlin ら [77] や Funge ら [35] によって報告されている．このように，運動学的手法は計算時間の面などで有利であり，実時間での再現が可能であるため，今日主流となっている手法である．

近年，映画や CM 制作等で普及しているモーション・キャプチャーも運動学的手法のひとつであり，順運動学的手法として位置付けられる．これは，人間の関節部分などに磁気センサなどを取り付け，その位置情報をある時間間隔ごとに測定して取得する接触式のものと，関節部分などに反射物体を装着して赤外線カメラなどで撮影してその反射物体の位置情報を取得する光学式のものに大別される．いずれも，現実の人間の動作データを実時間で取得できるという特徴があり，取得された動作データを仮想的な CG キャラクタに適用し，動きを実時間で再現することが可能である．そのようにして得られた動作データを，異なる人物像モデルに適用可能となるように修正する手法が Gleicher [39] によって提案されたり，Lee らによって複数フレームの人物像のデータを同時に表示して，対話的に動作の修正や変更を可能とした手法も報告されている [55]．このようにモーション・キャプチャーはかなり普及してきているが，機材やシステムが高価であることや，取得した位置データのキャリブレーションが面倒で誤差を生じやすいこと，撮影に広い空間やかなりの時間を要すること，取得した動作を他の動作に適用することはまだ難しいということなど，制約が大きいのが現状である．

一方，動力学的手法（dynamics）とは，関節角度や位置の変化が物理法則に従うものとして，運動方程式，即ち時間を変数とする微分方程式を立てて積分計算を行う手法であり，物理法則に基づくモデリング（physically-based modeling）のひとつとして位置付けられる．

図 2.4は，自然落下する球の運動を動力学的に生成した例を示す．このような運動を生成するには，以下のような処理を実行すればよい．

1) 初期位置 p，初期速度 v を決定する．
2) 加速度（鉛直下向に g）を決定する．
3) i = 0 とし，Δt を時間間隔として，以下の処理を繰り返す．
   3-1) 速度を計算する; $v = v + g\Delta t$.
   3-2) 位置を計算する; $p = p + v\Delta t$.
   3-3) その位置にある球を生成し，描画する．
   3-4) i = i + 1 とし，3-1) に戻る．

動力学的手法を適用することの長所は，この例のように，制作者が直接描画対象の位置や速度などを指定しなくても，何らかのパラメータ（初期位置や初期速度など）を指定することにより自然な動きを自動的に生成することが可能であるという点にある．

物理法則に従う物体を動かすには，力やトルクを加える必要がある．あらかじめどのような力や

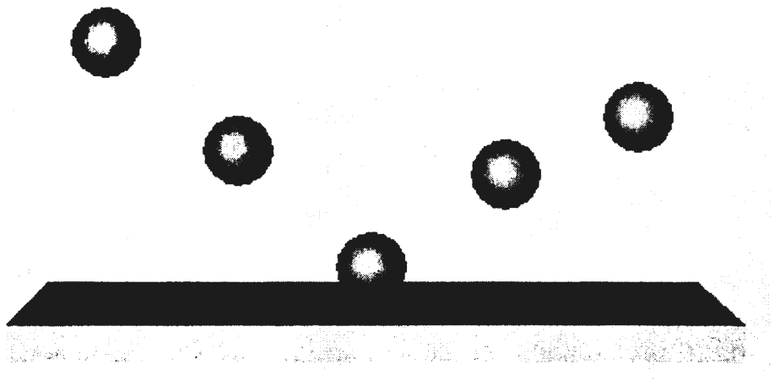


図 2.4: 動力学によるアニメーション生成例; 球の自然落下.

トルクを加えるのかが既知であり，順次運動方程式の積分を行っていく方法を順動力学（forward dynamics）と言う．逆に，ある時刻にある位置に達するにはどのような力を加えればいいか，を求めてから（あるいは求めながら）計算を行う手法を逆動力学（inverse dynamics）と言い，人物像の運動を動力学を用いて生成する場合は逆動力学的手法を用いなければならないことが多い．どのような初期値を与えれば望ましい時刻に望ましい位置に達するのか，という問題は一般的に逆問題と呼ばれている．

人物像に対し，上記の球の例のようなアルゴリズムを適用するには，人物像の動きがどのような方程式に従うのか，という定式化と，所望の動作を行うには，どのような力やトルクを加えれば良いか，という問題に取り組まなければならない．そのような力やトルクは，解が解析的に，かつ一意的に求まるということは一般的に保証されない．従って，逆運動学と同様に，何らかの拘束条件を加えて，多次元の連立一次方程式を数値的に解く問題に帰着することが多い．

人物像への動力学の適用に関しては，物体を構成する剛体リンクに対して運動方程式を適用する手法が Armstrong らによって提案され [3]，さらに人体モデルに拡張した手法が Armstrong ら [4][5] や Wilhelms ら [112][113] によって示された．一方，Barr，Barzel らは，人体のみならず，コンピュータ・グラフィックスにおける動力学的手法についての考察を行い，より一般的な，goal-directed な概念に基づく階層的な動作制御を行うアルゴリズムの構築を提唱した [10][11][12].

動力学的手法はリアルで自然に見える運動を生成することができるが，人体は多数の自由度を持つ複雑な物体であるため，完全な動力学による制御は定式化，計算コスト，数値的安定性の面から困難である．このため，多くの手法は動力学を部分的に用いるか，幾分限定されたモデルとして用いるなどして，計算時間を軽減しようとしている．このようにして生成された運動の例として，Ertl らの人物像の飛び込みシーンの生成 [29] や，Hodgins らのスポーツ選手の走行動作生成 [43] など

が報告されている．人体を構成する構造に対して何らかの拘束条件を課し，その拘束条件を満たしながら自然な動作を生成する手法も，比較的動力学の適用部分が少ない例が Badler ら [6]，Isaacs ら [46]，Lee ら [56] によって報告されており，また，動力学の適用部分が多い例も Rose ら [85] や Popović ら [80] よって報告されている．

このような運動の生成は，コンピュータ・グラフィックスのみならずロボティクス，医学，生体工学，生物学など広い分野にわたって研究されており，各種手法が提唱されている．ロボティクスの分野では早くからマニピュレータ等に対する制御を記述する手法が開発されており，Danavit - Hartenberg の表現などが有名である [31]．さらに，例えば Vukobratovic らに代表されるように，実際にロボットの手や足に対する力学系を構築する研究 [110][111] や，Brooks [17] による subsumption architecture のように，人工知能的な試みも行われている．また，ニューラルネットワークを適用することにより，制御の高速化をはかる手法も Grzeszczuk らによって提案されている [40]．医学や生体工学の分野でも，運動時の関節角度等の解析がなされており，藤田ら [32]，中村ら [67][68]，臨床歩行分析懇談会 [83]，Rose ら [86] によってまとめられている．

しかしながら自然な人物像動作を自動的に生成する決定的手法は未だ確立されておらず，発展途上の分野であるといえる．現在も，産業界ではキーフレーム及び逆運動学的手法によるアニメーション制作と，モーション・キャプチャーによって採取された運動データを適用する手法が中心的な手法であり，それを補助する形で動力学的手法や生体工学的な手法が採り入れられている．それにも関わらず，アニメーション制作の作業量をより削減する意味からも，自動的な動作生成を可能とする動力学的手法への期待は大きい．

2.4 歩行動作生成手法

歩行は人間の行動の中でもっとも基本的かつ重要なもののひとつであり，アニメーションシーンにおいて，歩行する人物像を加えることは非常に重要な事項である．特に歩行動作を自動的に生成することは，CG キャラクタの動作生成，3 次元環境モデルへの人物像の融合，設計や各種シミュレーションへの適用等の応用面において，リアリティあふれる表現を可能とするために極めて重要である [97]．人物像の歩行動作生成には様々な手法が提案されており，例えば Multon らによって [66] にまとめられている．

これらの手法を歩行動作の機能面から分類すると，まず，平面上を直進歩行する手法が提案され，次に平面上を曲線歩行する手法が提案された．その後，凹凸のある地形上の歩行動作生成が提案され，現在は個性を出すような歩き方や，あらかじめ生成した歩行動作を編集するような手法が報告されている．歩行動作の生成方法の面からは，伝統的な運動学的手法に基づくものから始まり，動力学的手法を部分的に用いる手法が現われ，さらに目的に応じて様々な手法が報告されている．

運動学的手法では，例えば，Phillips らによる逆運動学手法を用いる手法 [78]，Zeltzer の LMP（local motor program）を用いた手法 [116] や Calvert らのキーポーズ間の補間による手法 [24]，Boulic らの生体工学的データを適用する手法 [14] が報告されている．また，Overveld らは左右の

足の幾何的位置をあらかじめ決定し，運動学的に回転させることで方向転換を実現している [73].

モーション・キャプチャーによって採取した動作データベースを用いる手法では，鵜沼らの重み付けによって合成する手法 [108] や，Gleicher らの動作データを適用する対象に応じて，動作データを修正する手法 [39] などが報告されている．これらの手法は，対話的な動作生成が可能であるが，必要な動作データはあらかじめ採取あるいは生成しなければならないという問題がある．

動力学的手法を部分的に採り入れたものとしては，ロボティクスの分野では，現実の歩行ロボットを構築することを目指した多くの仕事がなされており，Raibert らによるジャンプ移動の生成 [81] など，いくつかの手法は CG にも適用されている．近年，実際に歩行ロボットを開発している事例も発表されている．動力学的手法はリアルで自然に見える運動を生成することができるが，人体の運動は自由度が大きく，計算時間が巨大にかかってしまう．従って，多くの手法は動力学を部分的に用いるか，幾分限定されたモデルとして用いるなどして，計算時間を軽減しようとしている．例えば，Bruderlin らは運動学的手法に動力学を取り入れ，少数のパラメータによる人物像の平地のみの歩行動作の自動生成を KLAW（Keyframe-Less Animation of Walking）システムで実現している [19][20][21]．Stewart ら [90] は *Newton* システムにおいて，直進歩行に対し，動力学的なアプローチを導入した．Laszlo ら [54] は limit cycle control を歩行やバランスに対するオープンループの挙動に適用し，Ko & Badler [52] は荷重や力が加わった歩行を逆動力学を用いて実現した．一方，動力学を適用する自由度を大きくし，立った姿勢や受動的な歩行を 90 もの自由度を持つ人体モデルでシミュレートした例も McKenna らによって報告されている [62].

これらの手法は主に平面上の歩行を前提とした歩行動作生成手法である．実際の CG シーンにおいては，様々な 3 次元の環境モデルが登場することが多く，また，今後もそのような 3 次元環境モデルが作られていくであろうと予想され得るため，地形に合致する歩行動作生成モデルが必要となっている．動力学的手法は複雑な運動の自動生成やシミュレーションなどに適しているが，定式化の難しさや計算コストなどの面から，複雑な地形などの現実の問題に適用できるような計算モデルは未だ提案されていない．逆に，CAD における建築物あるいは景観 CG などのように環境を生成することが主目的となっているシステムでは，動き回る人物像を全く使用していないか，使用していても環境とは無関係な動きを合成したものばかりであった．それらの背景から，地形に適合する歩行動作生成手法が提案され始めた．

地形に合致した歩行動作の生成は，歩行動作の生成自体に加え，地形情報の幾何的情報や地形情報から受ける制限，地形情報に対する相互作用など，さらに付加的な束縛条件の記述や制御モデルを必要とする．そのような，地形合致へのアプローチの一つは，自己適応アプローチである．例えば，McKenna ら [61] はセンサー & アクチュエータモデルを 6 足のモデルに適用し，凹凸ある地形上の移動シーンを生成した．van de Panne ら [74] は周期的な PD（proportional & derivative）コントローラを導入し，鉛直面内に限定されているものの，坂などの歩行動作を生成した．また，多賀らは神経回路網の振動子を歩行動作の制御に適用し，平地または傾斜の小さな坂に対する柔軟な歩行動作を生成している [91][92]．これらのアプローチは，適切な足の位置を見つけ，適切な歩行動作を計算するような行動モデルを構築することを目的としている．しかしながら，これらのアプローチは，何らかの境界条件をユーザが与えるわけではないため，キャラクタを所望のパス上を動かすように制御するのは困難である．

もうひとつのアプローチはフットプリント（足跡）・アプローチである．これは，ユーザの望ましい移動経路に沿った歩行を実現するため，足の位置あるいは歩行パスがまず決められ，与えられた足跡や歩行パスに合致するような歩行動作が計算されるものである．運動学的な手法においては，Girard [38] や Overveld ら [73] は足の位置の間の移動方法を報告し，Bezault ら [13] は歩行の軌跡を対話的に制御する手法を提案している．Ko ら [51] は，人体重心位置と直面している方向から次の足跡位置を評価し，そこへ人体を移動させる *VRLOCO* システムを導入し，平面上での様々な種類の移動を実現している．一方，van de Panne [75] や Torkos [95] は動力学的な手法を用いた足跡間の移動を実現している．この手法は比較的制作者の望むような移動経路に沿った歩行動作を生成することが可能である，という特徴があるが，歩数が増加すると足跡の設定自体に時間がかかってしまうという問題がある．

以上，多くの歩行動作の生成手法が報告されているが，従来の手法の多くは，階段などを含む複雑な地形を含む環境コンテンツに対する歩行動作の生成は困難であるか，あるいは作業量が増大しまうという問題点があった．

2.5 本研究の位置付け

近年のコンピュータ・グラフィックス技術は高品質な人間の歩行動作を生成し，それをキャラクタ・アニメーションや動作シミュレーションに適用することを可能としているが，動いている人物像を仮想的な環境に適合させるのは非常に困難で時間を要する．その理由として，環境やオブジェクトと人物像との相互関係を記述することは難しいことなどが挙げられる．また，人間の運動の生成においては，適切なリアリズムの達成にはユーザからの多大な人力を要する．

しかしながら，広く動作生成を考慮していくにあたり，CG キャラクタの仮想世界における環境あるいはオブジェクトとの関係は重要な要素であり，今後そのようなシーンへの需要が増大していくであろう．本研究は，3 次元仮想空間に応じた人物全身像の動作を自動的に生成するアニメーション手法の実現において，人物像の自然な歩行動作を，できるだけ少ないパラメータにより制御でき，かつ，計算量が少なくて済むアルゴリズムの構築を目的としている．

仮想環境との関わりとして，本研究では既知の静的な環境としての地形データ（幾何的データ）への合致を取り上げる．これは，凹凸ある地形に合致した歩行動作を自動的に生成する手法の範疇である．即ち，メモリ上に読み込まれている 3 次元の地形データ上に歩行パスを設定し，そのパスに沿うように歩行動作を生成するパス・アプローチに基づいている．その際，あくまでアニメーションの制作者が意図する動作を生成する，という目的から，歩行パスは CG キャラクタが自動的に生成するのではなく，ユーザが設定することを前提とする．本研究で述べる手法は平地歩行や階段の上昇・下降，方向転換を含む高度に制御可能な人間の動きを達成し，複雑なシーンを作成するのに

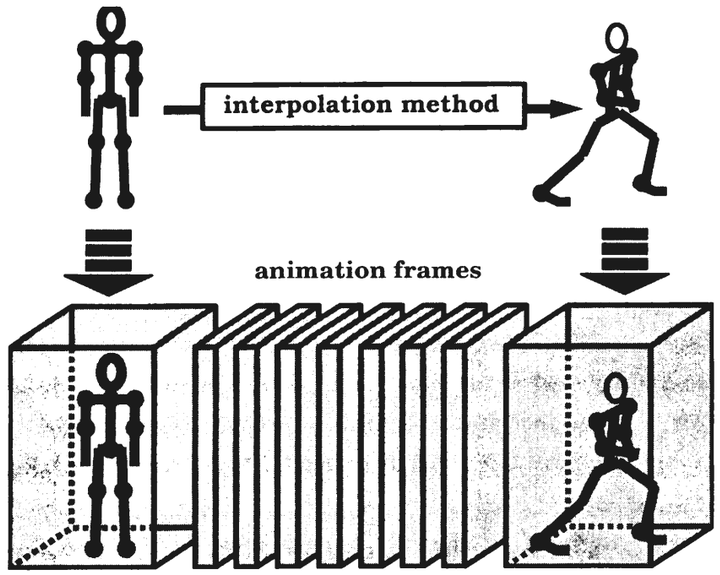


図 2.5: 本研究における人物像の動き生成の概念.

要求される作業コストおよび計算時間を削減する．環境に適応できる歩行動作を自動的に生成することにより，建築物等の CG に歩行する人物像を加えて現実感を出したりなど，幅広い応用面が期待できるとともに，映像制作におけるユーザの負担を軽減できる．

本研究は，図 2.5に示すように，ある拘束条件に基づいて，所定の時刻に所定の姿勢になるような境界値を計算し，運動学的な計算と動力学的な計算を融合した手法を用いて自然な動きを生成するように補間する手法のひとつである [99]．動作生成の段階で，平地歩行や階段上昇，階段下降，曲線経路に沿った方向転換を含む自然な歩行動作を，どのように定式化するか，という問題と，動力学を用いる部分にどのような力やトルクを加えたら良いか，という逆問題への解決をはかるものである．

本研究では，まず，平地直線歩行動作生成手法を発展させ，高低を有する地形上における人物像の直線歩行動作生成法を実現する．次に，その生成法を前進歩行モジュールとし，新たに左右への方向転換を行うモジュールを追加して両者を結合させ，3 次元形状を有する地形に沿う歩行動作の生成を実現する．次に，これらの動作生成の動力学部分の運動方程式に対し，床反力を近似した関数を外力として用いることにより，自然かつ高速な動作生成を実現する．そして，これらの動作生成を用いた人物像歩行動作を多数生成し描画する際に，視点からの距離に応じて運動の更新周期を変化させることによって，より効率的な生成・表示を実現する．さらに，これらの手法を適用したアニメーションシステムを実際に構築し，様々な映像コンテンツ制作に適用することにより，その有効性を確認する．

まず，本研究での基本となる直線歩行動作生成手法は，平地直線歩行を生成した Bruderlin の KLAW（Keyframe-Less Animation of Walking）システム [19] を発展させ，高低を有する地形上での人物像の直線歩行動作生成法を実現したものである [97][98][99]．KLAW システムは，

(1) モデルを単純化することにより，運動方程式の適用を容易ならしめている，
(2) 立脚と遊脚とを分離して考察し，立脚に第 3.4で説明する virtual leg concept を導入することにより運動方程式の未知数を減らしている，
(3) 少数パラメータにより，自動的に歩行動作が生成できる，

などの長所があり，自然な平地直線歩行動作の生成を可能としている．

その一方で，

(1) 運動学的要素の多くが観察データからの引用であり，動作生成の一般性が低い，
(2) 力やトルクの計算に時間がかかる，
(3) 遊脚の軌道が 4 次関数で近似されており動きの制約が大きい，
(4) 計算が鉛直面内で実行されているため，3 次元地形変化への自動的な適応が困難である，

などの問題点もある．そこで，3 次元地形を有する仮想環境に合致した歩行動作を生成するために，凹凸ある地形を，左右への方向転換を行いながら歩行することが可能なモデルの構築を試みる．これは，3 次元仮想環境に設定された 3 次元歩行パスを直進部分と回転部分に分解し，一歩の歩行動作を前進方向と回転方向とに分解して，それぞれが歩行パスの直進部分および回転部分に沿うようにそれぞれに動力学及び運動学を適用して動作を生成し，関節角度の可動範囲や体軸の傾きを変更することで地形あるいは歩行パスに適応した歩行動作表現の自動生成を可能とするものである [101]．

これにより，これまでのシステムでは不可能であった環境コンテンツに対する歩行パスの設定のみで地形に適応した歩行動作を高速に生成することが可能となった．これらの手法については，第 3 章で詳しく説明する．

次に本研究では，さらに計算効率の良い歩行動作の生成を実現するために，個々の歩行者の動作生成を高速かつ安定に行い，容易に制御するための手法を導入し，さらに，複数の歩行動作を生成する際の計算コストの削減を試みる．

まず，歩行動作を生成する運動方程式の外力項に床反力を適用することで，より自然で安定した計算を実現する．前述の動作生成手法においては，外力として歩行動作を解析した文献から引用したトルク関数を用いており，歩幅が小さいときに数値的に不安定となっていたが，床反力を測定し，解析関数でフーリエ近似した関数を外力として適用することで，数知的な不安定性がなくなり，より安定かつ高速な計算が可能となった [107]．また，これらの手法を用いて生成した歩行動作を多数の人数にわたって生成し描画するための手法 [105] についても言及する．これは，視点から遠くの人物像は相対的に移動量が小さい，ということを利用して，視点からの距離に応じて歩行動作の運動更新の周期を変化させ，視点から近い人物像は毎フレームごとに運動更新を行い，視点から遠い

人物像はいくつかのフレームごとに運動更新を行うものである．これらの手法については，第 4 章で詳しく説明する．

さらに，本研究では，これらの手法を用いたアニメーション・システムを実際に構築し，背景画像と 3 次元歩行アニメーションとの合成や人物画像を用いた歩行動作アニメーションの生成，ネットワークを介したアニメーションの制作などに適用した例を示す．このように，実際の映像コンテンツ制作に適用することによって，本研究の有効性を確認することができる．ソフトウエア・システムの構築やこれらの適用例については，第 5 章で詳しく説明する．

第 3 章

地形に沿う人物像歩行動作の生成

3.1 はじめに

本章では，3 次元仮想空間内の，任意の 3 次元地形に設定された歩行経路（歩行パス）に沿う人物像の歩行動作を自動生成する手法について述べる．

本章で述べる手法は，従来，平面上に限定されていたり，周囲の環境とは無関係であった歩行動作を，階段などの上下方向の変化や，左右への方向転換を含む 3 次元歩行動作に拡張することにより，アニメーションを制作する利用者が望む経路に沿った歩行動作を容易に生成することを可能とする．

本手法の特徴は，歩行経路を一歩ずつに分割して空間的な境界条件を計算し，歩行動作を互いに動力学的に独立な直進歩行動作，腕の運動，回転運動に分解して運動を生成する方程式を立て，歩行動作を生成する点にある．

本章では，まず，本論文にわたって用いられる人物像の論理的構造について説明を行い，ついで歩行動作に関する境界条件について説明を行う．その後，平面上の直進歩行動作を拡張し，上下方向の地形変化に合致するような歩行動作生成を実現する手法について述べる．次に，腕の運動を生成するための手法に付いて述べ，さらに，左右への方向転換を含む 3 次元歩行パスに合致する歩行動作の生成を可能とする手法について述べる．

本章は本研究の根幹をなすものであり，本章の成果により，3 次元仮想環境に移動経路を設定するだけで，移動経路に沿う歩行動作を自動的に生成することが可能となる．

3.2 人体の構造

この節では，本論文において共通して用いられる，3 次元仮想環境における人物像モデルの関節構造および自由度，座標系などの定義および用語について解説する．本論文においては，人体を構成する論理的モデルは単純な剛体スケルトン棒で近似され，動力学的な運動は，剛体棒に対する運動方程式を反復計算することによって生成される．

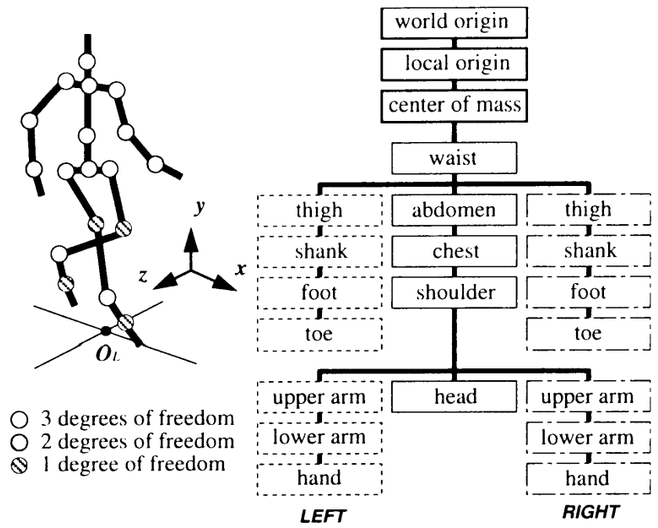


図 3.1: 多関節構造体としての人物像モデルの構造.

本論文では，座標系は全て右手系で表すものとする．また，計算機内で表示しようとしている仮想的な 3 次元空間に設定されたグローバルな座標系を世界座標系またはワールド座標系と呼び，この世界座標系においては，コンピュータ・グラフィックスでの慣習に基づき表示画面に向かって右方向を＋X 軸に，上方向を＋Y 軸に，表示画面から観察者に向かう方向を＋Z 軸に取るものとする．

本論文で取り扱う人体の論理的な構造は以下のようなものである．

基本的に人体の各部位を長さが一定で質量を有する剛体線分で近似し，各線分の端点を点（関節に相当する）で接続することにより，仮想的な多関節構造体を構成する．

本論文における人体の構造を図 3.1に示す．人体はツリー状の階層構造をなし，各部位は関節によって接続されている．本論文では腰の中心を人体中心即ち原点とし，以下，図 3.1に従って全体を構成していく．図 3.1からわかる通り，人体においては，各部位に対し，必ず親となる部位がただ一つだけ存在している．ここで言う「親」とは，ある部位に接続している他の部位のうち，より原点に近い方を指す．

本論文では，ある部位が親となる部位に対してどれだけの角度で接続されているか，によって一意的に姿勢を記述する．即ち，世界座標系の座標軸に対する回転角度や位置で表すのではなく，各部位がその部位に固定されたローカルな座標系を有し，各座標系が親となる部位の座標系からどれだけ変換されているかでその部位の姿勢を決定する．

本論文では人体の各部位を長さが変化しない剛体棒で近似しているため，親となる部位からの座標変換は，回転角度のみで表すことが可能である．従って，各時刻における人体の位置や姿勢は，

表 3.1: 人物像モデルを構成するデータ（数字は自由度）

	共通		立脚		遊脚	
位置	重心位置	3	足首位置	3	--	
角度	頭部	3	足部関節	1	足部関節	1
	上体	3	足首関節	3	足首関節	3
	骨盤	3	膝関節	1	膝関節	1
	肩骨	2	股関節	3	股関節	3
			肩関節	3	肩関節	3
			肘関節	2	肘関節	2
			手首関節	2	手首関節	2

その時刻における人体中心の位置および向きと，各関節角度を決定することによって決定することができる．これらの物理量（人体中心位置，向き，各関節各度）を時間軸に沿って生成することが運動生成の目的となる．本論文では，動作生成の計算時に，脚を立脚と遊脚に分て計算を行う（立脚，遊脚については第 3.3.1 節で述べる）．表 3.1 に，立脚側，遊脚側，共通部分の各関節の自由度を示す．

人体はこのように多くの自由度を有する複雑な構造体である．これら全ての関節角度を変数として運動方程式を立てて計算を行うと，多大な計算時間がかかってしまうことになる．既に述べた通り，本研究は，動力学と運動学を融合した手法により，歩行動作を生成する．即ち，図 3.2に示すように，人体のおおまかな部分を，後述する境界条件を満たすように動力学により生成し，残りの部分を運動学により補間する．

実際に計算機内で歩行動作を生成する際，動力学計算においては，各部位の質量や長さが必要となり，運動学計算においても，各部位の長さを必要とする．本研究では，各部位の質量や長さをあらかじめ指定しておく必要があるが，人体の密度が一定であるという仮定のもとに，人体の各部位の質量および長さの，それぞれ体重および身長に対する標準的な比率を用いて，身長と体重のみの指定からでも計算を可能としている．これらの比率は，人体計測の結果をまとめた文献，例えば [32][49][69][70] などによって得ることができる．また，歩行時の主要関節角度の可動範囲即ち下限値・上限値も，歩行を分析した文献 [67][68][83][86]，などに示された範囲から標準的な値を指定することができる．

3.3 歩行に関する準備

この節では，歩行動作に関する一般的な用語を解説し，歩行動作を行う際の時間的な境界条件，空間的（幾何的）な境界条件について説明する．

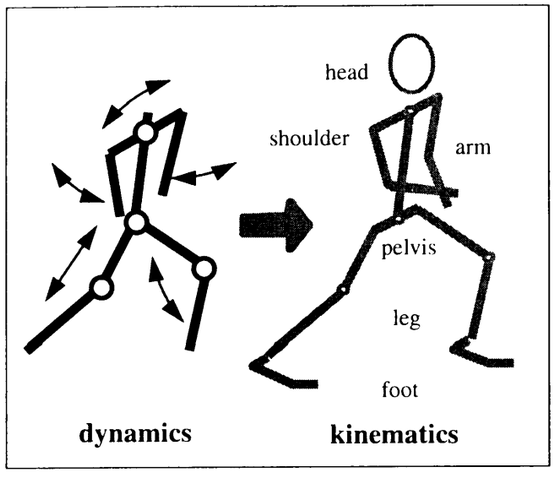


図 3.2: 動作生成の基本的な考え方; 動力学でおおまかな運動を生成し, 運動学で調整を行う.

3.3.1 歩行時の持続時間

ここでは，本論文で用いられる歩行の持続時間について解説する．本論文において，ステップ（一歩）とは，片方の足が地面に着地（heel strike）してから他方の足が地面に着地するまでを表す．ステップというときに，この一連の動作自体を指すこともあれば，時間間隔または持続時間を表すこともある．左右の足が一歩ずつ歩行する時間間隔を 1 サイクルと言うことがある．

片方の足が地面に着地してから離れる（toe off）までの時間間隔を**立脚相**といい，地面に接地している方の脚を**立脚**（stance leg）と言う．片方の足が地面から離れてから再び着地するまでの時間間隔を**遊脚相**といい，地面から離れている方の脚を**遊脚**（swing leg）と言う．また，両方の脚が地面に接している状態を**両足支持相**（double support）という．

ここで，歩行の時間間隔または持続時間（duration）を記号を用いて表すことにする．ステップの時間間隔を T，立脚相の時間間隔を T^{stance}，遊脚相の時間間隔を T^{swing}，両足支持相の時間間隔を T^{ds} とすると，以下の関係式が成り立つ：

$$T^{stance} = T + T^{ds}, \tag{3.1}$$

$$T^{swing} = T - T^{ds}, \tag{3.2}$$

以上の関係を図 3.3 に示す．なお，T は，ステップの持続時間であることを強調するために T^{step} と記すこともある．また，各ステップに順序を表す識別子がついているとき，例えば第 i 歩であることを示すときは，T_i のように，添字付きで記すものとする．歩行時には，腕の運動もこの持続時間に従う．第 3 章で詳述するが，一般に立脚側の腕は後方から前方へ振り出され，遊脚側の腕は前方から後方へ振り戻される．

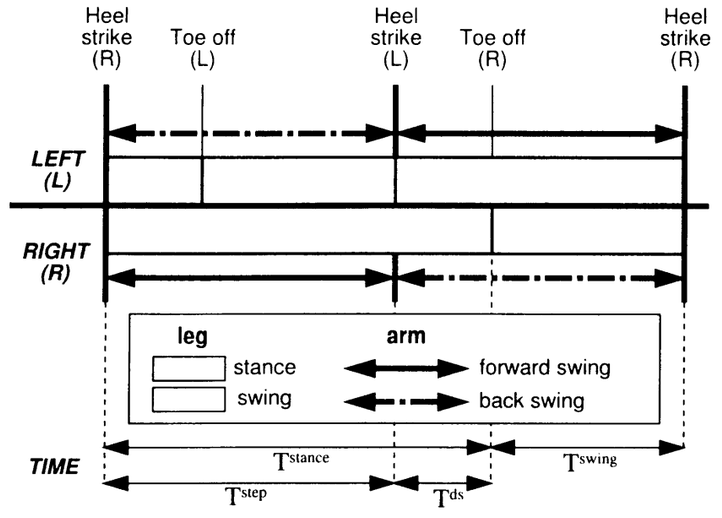


図 3.3: 歩行に関する持続時間.

3.3.2 歩行の幾何的境界条件

本論文は一歩（ステップ）ごとの境界条件に合致するように運動の生成を行うアプローチを取る．従って，ステップに関する境界条件を求める必要がある．即ち，歩行速度，歩幅，地面の高低差，方向転換を行うときの身体の回転角度などである．

図 3.4 はこれらの幾何的な境界条件を示す．図 3.4 において，$\boldsymbol{P}_i$ は i 番目の踵着地点を表している．この時，同じ水平座標を持ち，高さが異る点がある場合は，高い点のみが用いられる．w_i は $\boldsymbol{P}_i$ と $\boldsymbol{P}_{i+1}$ の間の歩幅すなわち水平距離であり，h_i は $\boldsymbol{P}_i$ と $\boldsymbol{P}_{i+1}$ の間の高低差すなわち鉛直距離である．α_i は回転角度，即ち，$\boldsymbol{P}_{i-1}$ と $\boldsymbol{P}_i$ を結ぶ直線の延長線 $\boldsymbol{P}_i\boldsymbol{P}'_{i+1}$ と，$\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ とが，**水平面上**でなす角度を表している．以後，本論文では，高低差を歩高と呼ぶことがある．

一歩の間に，人物像モデルは $P_{i-1}P_iP_{i+1}$ という運動を生成する．その際，身体の向きを $\vec{\boldsymbol{d}}_{i-1}$ $(= \overrightarrow{P_{i-1}P_i})$ から $\vec{\boldsymbol{d}}_i (= \overrightarrow{P_iP_{i+1}})$ に回転させることになる．このように一歩の歩行動作を生成するための境界値となる $\{ w_{i-1}, h_{i-1}, w_i, h_i, \alpha_i \}$ を本論文では幾何的パラメータと呼ぶことにする．

また，持続時間と同様に，ステップの順序を表す識別子が付いているときは，図 3.4 に示すように w_i，h_i，α_i のように添字付きで示すものとするが，混乱を生じないときは単に w，h，α のように示すことがある．この場合，前節で定義したように，一歩は「踵着地をもって」始まるものとしてあるため，「現在のステップ」は，現在の立脚の踵が着地した時点に始まり，次の踵着地，即ち現在の遊脚が踵着地を行う時点で終了する．

さて，歩行における水平面上の移動速度を v とすると，前節で述べた ステップの持続時間 T の値は，v と歩幅 w とから

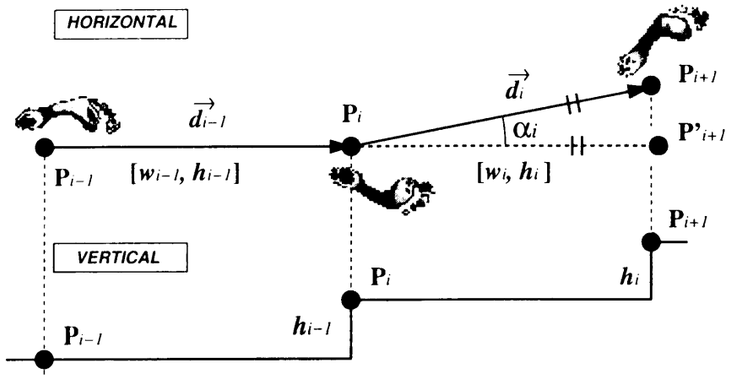


図 3.4: 歩行時の境界値.

$$T = \frac{w}{v}, \tag{3.3}$$

により容易に得ることができる．この式は，移動速度と歩幅とを決定すれば，ステップの持続時間が決定できることを示している．

動作の計算においては，各関節角度や位置に関する境界条件も必要であるがこれに関しては後述する．

3.3.3 歩行時のパラメータ

本論文においては，3.3.1節，3.3.2節で述べた境界条件を満足するように，各ステップの境界間を自然な動作を生成するように運動を計算するが，人物像モデルの各部位の長さや質量が与えられたとき，歩幅，高さ，回転角度，移動速度により，ほぼこれらの境界条件は決定づけられる．移動速度は環境モデルや人物像には依存しないが，歩幅は平面上の歩行時においては人物像モデルに依存し，高低さある地形の歩行時においては高さとともに地形モデルに依存する．また，回転角度は移動経路に依存する．これらの { 歩幅，歩高，回転角度，移動速度 } をまとめて歩行パラメータと称することがある．

本研究においては，これらの歩行パラメータを基に境界条件を計算し，歩行動作に関する運動方程式を立て，あるステップの開始時刻から T 後に境界条件に合致するように動力学計算を行う．境界条件に合致しない時は，運動方程式に現れる物理変数，具体的には一般化力の値を変化させながら反復して計算を行い，境界条件との差が一定の閾値以内におさまるまで繰り返す．最終的に単位描画時間毎の関節角度の時系列データを計算し，画像としての表示を行う．この計算方法は，本論文で述べる全ての計算に適用されているものである．

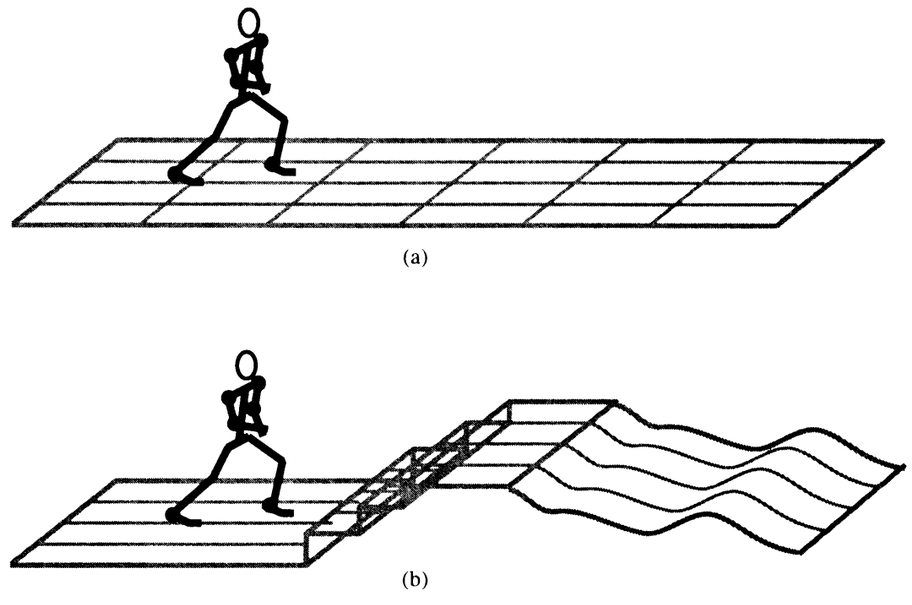


図 3.5: 高低差ある地形への適用; (a) 平地, (b) 複雑な地形.

3.4 直線歩行に対する人物像歩行動作の生成

コンピュータ・アニメーションにおける歩行動作生成の最も基本となるものは，平地直進歩行動作生成，即ち高低差のない，理想的な水平面上における直進歩行動作の生成手法である．なぜならば，計算の次元を事実上鉛直面内に限定することが可能であり，また，人物像モデルによる差異はあるものの，各ステップごとの幾何的な境界値が，ほぼ歩幅のみで決定できるからである．しかし，これらの生成法では，あくまで人物像中心の記述とならざるを得ないものが多く，環境オブジェクトは後から付け加えられたアクセント的な役割に過ぎないものが多い．生成手法に物理モデルに基く手法が採り入れられることにより，衝突や接触などの表現が可能となったものの，環境オブジェクトはあくまで人物像動作の対象に過ぎず，環境オブジェクト自体が幾何的あるいは物理的特性を備え，そこを歩き回る人物像に対して能動的に影響を及ぼすことは，困難である．

本節では，高低差ある直線経路に対する歩行動作生成手法について述べる．本節で述べる手法の目的は，図3.5あるいは図3.8に示すように，高低差ある地形に合致した人物像の歩行動作を生成することであり，「自然な歩行動作」をできるだけ少ないパラメータにより制御できるアルゴリズムを構築することである．坂道や階段昇降などの，現実世界に存在する環境に合致した歩行動作を自動的に生成することにより，例えば建築物等の CG に歩行する人物像を加えて現実感を出したり，アニメーションへの幅広い応用面が期待できるとともに，アニメータへの負担の軽減が実現できる．

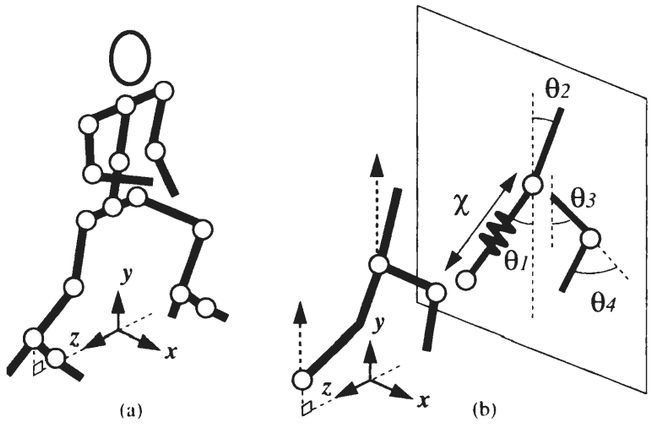


図 3.6: 人物像モデルと動力学変数 (a) 人物像モデル, (b) 座標系と動力学変数.

本節で述べる手法は，平地歩行動作生成手法を拡張し，高低差ある地形に対応可能な直進歩行動作の生成を実現する．基本となる平地歩行動作生成手法に関しては，物理法則を用い，かつ少数パラメータで制御できる特徴があるという理由で Bruderlin の KLAW （ Keyframe–Less Animation of Walking ）システム [19] を用いる．この手法の特徴としては，

- モデルの単純化により運動方程式の直観的適用を容易にしている，
- 立脚と遊脚とを分離して考察し，立脚に virtual leg concept を導入することにより運動方程式の未知数を減らしている，
- 少数パラメータにより，自動的に歩行動作が生成できる；

などの長所がある一方で，

- 歩行動作生成が理想的な平面上に限定されている，
- virtual leg や遊脚軌道の生成時に，数値計算上の不安定さを生じることがある，

などの問題点もある．ここで virtual leg とは，足首と腰中心部とを結ぶ仮想的な線分のことであり，図 3.6 (b) や図 3.7 の χ に相当する．

本節で述べる手法では，上記問題点を解決し，他の歩行動作へも適用できるように，新たに
(1) 地形に沿った歩行パスの，ステップ毎への分割，
(2) 歩幅，速度のみであった歩行パラメータに，新たに高さを加えた境界条件に対する運動の生成，
(3) 2 つの 3 次関数の組合せによる遊脚足首軌道の生成，
を適用し，これによって，高低差ある地形に対する直進歩行動作の生成を可能とした．

本節で述べる手法により，人物像歩行動作のアニメーション生成において，ユーザが地形に合わせて歩行動作の調節を行なう必要は全くなく，地形とそれに対する直線歩行パスを入力することで，それに合わせた歩行動作シーケンスを自動的に得ることができる．

以下，本節では，歩行動作生成に必要な動力学変数について述べ，歩行動作生成の運動方程式を導く．次に，本節の内容をインプリメントする際の手順について述べ，高低差ある地形に適用するための境界条件を導く．その後，動作生成について詳しく述べ，生成例を示して検討を行う．本節の内容は，本節以降の全ての章の基本となるものである．

3.4.1 動力学変数および運動方程式

第 2 章でも述べたように，人間のように多くの自由度を持つ複雑な構造体の動きを表現するにあたって，代表的な動力学変数のみを抽出し，運動方程式を用いて境界条件に合致する運動を生成する．その後，残りの関節角度を，動力学変数に合わせて，境界条件に合致するように補間する．

図 3.6 は，動力学を用いる座標系と動力学変数を示している．本論文では，図 3.6(a) に示すように，$+X$ 軸方向に人物像が直進するものとし，上向きに $+Y$ 軸を取っている．

本節では，動力学を，図 3.6(b) に示す 5 つの変数に対して適用する．即ち，動力学を適用する人体モデルを，上半身，立脚部，遊脚大腿部，遊脚下腿部の 5 つのパーツに分け，各パーツを点で連結する．図において，χ は立脚足首と腰とを結ぶ長さ（立脚長）であり，θ_1 は立脚部が鉛直線となす角度（立脚股関節角度）であり，θ_2 は上体部が鉛直線となす角度（上体角度）であり，θ_3 は遊脚大腿部が鉛直線となす角度（遊脚股関節角度）であり，θ_4 は遊脚下腿部が遊脚大腿部となす角度（遊脚膝角度）である．これらの変数は全て YZ 平面内で定義される．立脚は伸縮する棒で表され，縮んだ場合は膝の屈曲を，伸びた場合は膝の伸展及び踵の上昇を表す．遊脚膝角度（θ_4）は運動方程式に含まれるが，逆動力学に付随する問題，例えば解が一意的に決まらない，といった問題を避けるために，あらかじめ足首軌道曲線を決定することによって運動学的に計算する手法を採用する [19].

図 3.7 に，鉛直面内の動力学変数と長さを示す．図 3.7 において，動力学変数は図 3.6と同様であるが，l_2 は上体を近似した剛体棒の長さ，l_3 は大腿部を近似した剛体棒の長さ，l_4 は下腿部を近似した剛体棒の長さ，l_1 は l_3 と l_4 との和であり，χ が l_1 を越えるときは踵が上昇していることを示す．計算時には，立脚足首の位置を固定した原点とみなし，また，式の複雑さを避けて計算時間を軽減するために遊脚部分を分離して [18][19]，Lagrange の運動方程式を適用する．方程式に関する詳細は，第 3.4.5節で後述する．

3.4.2 歩行動作生成の手順

この節では，本節で述べる歩行動作生成を実装する際の手順について述べる．本節での手順は，基本的に，後述する第 5 章の図 5.3に従うものであり，以下のようになる：

1. 人物像のデータの設定，

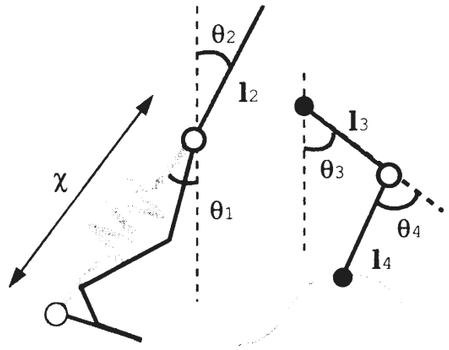


図 3.7: 動力学変数.

2. 環境データの読み込み，および環境データへのパス設定，
3. パスの分割および境界条件の計算，
4. 動力学計算，
5. 運動学計算，
6. 表示.

まず，人物像のデータを設定する．具体的には，身長や体重，標準歩幅や平均歩行速度，運動学で補間する関節角度などの可動範囲，などである．これらの設定値は，動力学計算時の定数の計算や，境界条件の計算に用いられる．

次に，環境としての地形データを読み込む．地形データは，階段や坂道などの地形を構成する幾何的な情報が，仮想空間内の 3 次元座標で完全に把握できるものとする．ユーザは例えばグラフィカル・ユーザインタフェースにより，対話的に歩行パスを設定する．本節では，出発点と停止点の 2 点を決定することにより，この 2 点を結ぶ直線に沿って，地形上を歩行する．

次に，前述の歩行パスを，ステップ（一歩）毎に分割する．その際，歩行パスに対応する直線と 3 次元環境データとの交点や，標準的な歩幅の値などが分割の判断材料として用いられる．歩幅に対しては，標準的なデフォルト値が設定されているためユーザは特に指定する必要はないが，変更することも可能である．この段階で，入力された地形変化に対応した適切なステップの列を記述し，各ステップに対して適切な歩行動作（上昇／平面移動／下降）を設定する．同時に，歩行動作の計算に対する境界条件を計算する．この段階によって，地形情報の入力およびパスの設定のみで地形に応じた歩行シーケンスの自動的な生成が可能となる．

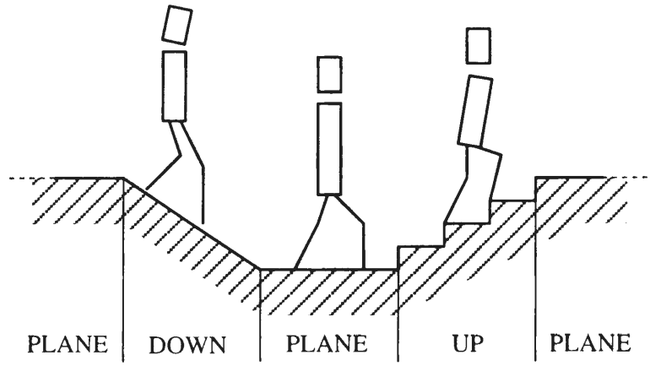


図 3.8: 地形への適応例.

次に，前述の境界条件を運動方程式に適用し，アニメーションにおける各フレーム毎の人物像の位置や方向，関節角度を生成する。その際，第 3.4.1 節で述べた動力学変数に対して運動方程式を解くことにより，境界条件を満たす運動の生成を行い，さらに，他の部分の関節角度や位置情報を補間する.

その後，生成した人物像の重心位置や関節角度を人物像のキャラクタモデルに適用し，環境モデルと併せて，歩行動作の 3 次元画像シーケンスを各フレーム毎にディスプレイ上に表示する.

3.4.3 歩行パスの分割

本節では，高低差ある地形に設定された直線歩行パスを，適切な一歩ごとの列に分割する．まず，環境モデルに設定された直線歩行パスに対し，“エリア分割” および “ステップ分割” を適用して各種の境界条件を有する「ステップ」オブジェクトを生成する．なお，本論文では，“エリア” という言葉は，歩行パスにおいて同じ高さを持つライン・セグメントで，幾何的な情報を付与されたオブジェクトとして用いている.

エリア分割

入力された直線歩行パスを，“エリア” に分割する．エリア分割は以下の手順で行われる：
(1) パスを水平線分と鉛直線分の組合せで近似する.

まず，与えられた環境モデルに対し，出発点と停止点を入力する．その断面図は図 3.9(a) のようになっている．その後，パスを同じ幅を持つ小区間に分割し，区間内で高さの平均値を計算する（図 3.9(b)）．隣合う区間で高さの平均値がある閾値を越えないような場合は，それらをまとめて 1 つの区間とみなし，2 つの区間の平均値をもってまとめた区間の高さとする．その後，図 3.9(c) のように，水平な線分と鉛直な線分からなるパスに近似し，近似パスを構成する制御点（連結点と

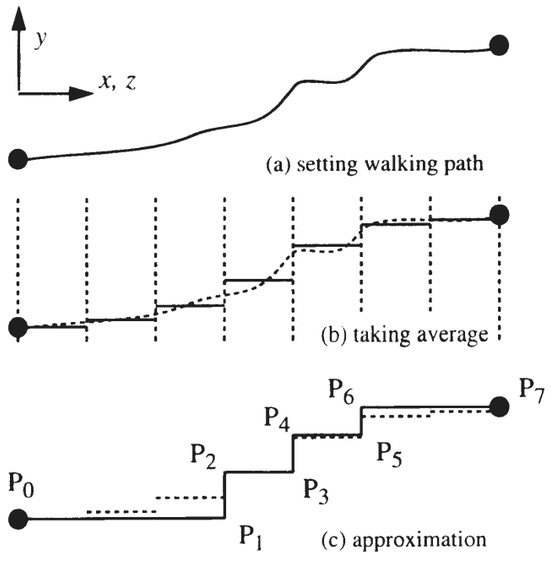


図 3.9: 歩行パスの近似.

始点, 終点；例えば図 3.9(c) の $P_i, i = 0, 1, \ldots, 7$) 座標値を決定する．区間幅や高さの差の閾値を適切に選ぶことによって，十分実用的な近似が得られる．

(2) 各区間に情報を付与する．

ここでは，まず，各区間の長さ（水平距離）によって，“平地” あるいは “階段” のいずれかの “地形情報” を与える．標準設定では，人物像モデルの標準歩幅（標準的な人物像モデルで 0.7 m 程度）を越えると “平地” とみなし，それ以下の場合は “階段” とみなしている．“エリア分割” の結果として，図 3.10に示されているように各区間 A_i に情報 g_i^A，即ちエリアの水平長さ（d_i^A），高さ（h_i^A），地形情報（I_i^A），直前のエリアの地形情報（I_{i-1}^A），次のエリアの地形情報（I_{i+1}^A）が付与される．高さ h_i^A は A_i と A_{i+1} との相対的な高さの差であり，歩行可能な高さの閾値を越えるような場合は，この段階でそのパスは歩行不可能としてリジェクトされる．

ステップ分割

“エリア分割” 終了後は “ステップ分割” が実行される．図 3.10に示すように，各ステップ S_j には情報 g_j^S，即ち歩幅（w_j^S），歩高（h_j^S），速度（v_j^S），直前のステップの “地形情報”（I_{j-1}^S），そのステップの “動作情報”（M_j^S），次のステップの “地形情報”（I_{j+1}^S）が以下のようにして付与される．この段階での速度は，各種設定段階で設定された，平均移動速度である．

まず，ある A_i での地形情報 I_i^A が “平面” の場合は，ステップを基本的に標準歩幅ごとに区切るが，エリアの水平距離 d_i^A や前後の地形情報などによって適当に調整し，A_i 内での歩数及び歩幅

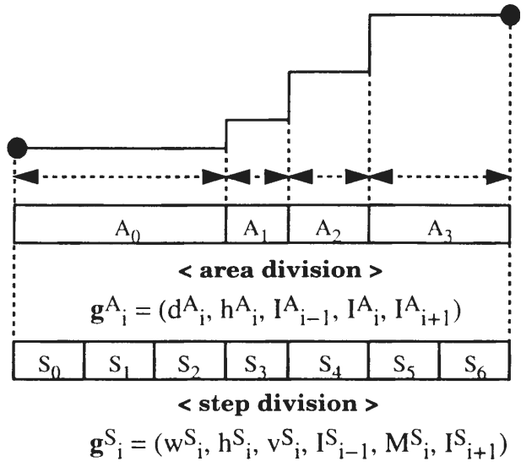


図 3.10: エリア分割とステップ分割.

の組合せ $(w_{i_0}^S, w_{i_1}^S, \ldots)$ を決定する。A_i での 1 歩めの歩高は h_{i-1}^A に等しく，最後の歩高は h_i^A に等しい．その間の歩高は全てゼロである．“階段” であればそのエリアでのステップ数はもちろん 1 であり，d_i^A，h_i^A がそのまま歩幅，歩高となる．このようにして出発から停止までの全ての歩数が決定される．

次に，地形情報を決定する．地形情報 I_j^S は，全てそのステップが属するエリアの地形情報 I_j^A に等しい．

次に，動作情報を決定する．動作情報 M_j^S は “階段から平地”，“階段から階段”，“平地から平地” などの歩行動作に関する情報で，I_{i-1}^S 及び I_{i+1}^S に依存する．

なお，前後の地形などを付与するのは，各ステップの計算における独立性を保つためである．このようにして，主に地形に関する幾何的な情報が付与されたステップ・オブジェクトが計算され，次の段階として歩行動作生成に必要な境界条件が計算される．

3.4.4 境界条件の計算

前節で計算されたステップ情報 g_j^S に基づき，適切な歩行動作生成のための境界条件の計算を行う．境界条件には以下の 4 つがある：

(1) 動力学変数に対する初期・終期条件（踵着地での値），

(2) ステップの持続時間，

(3) 遊脚足首の軌道（次節で述べる），
(4) 関節角度の可動範囲.
動力学的変数の初期・終期条件は，歩幅（w），歩高（h）を用いて，踵着地時（図 3.11）の関節角度などを三角関数などを用いて計算できる．ただし，踵着地の瞬間，腰は歩幅の中点の真上に位置するものと仮定している．これは，両足に均等に体重がかかっている瞬間とみなしたためであるが，歩高やその他条件に依存するものと考えられるため，今後改良の余地がある．例えば図 3.11に示すような階段上昇の場合，歩幅と歩高をそれぞれ w，h とし，腰の高さを H_Y とすると，χ_f は l_1 に等しい．図において，添字 $_{cur}$ は現在のステップを，添字 $_{pre}$ は直前のステップをそれぞれ示し，また，添字 $_{in}$，$_f$ は初期値，終期値をそれぞれ示す.

このとき，

$$\chi_{in} = \sqrt{(H_{Ypre} - l_{ankle} - h_{pre})^2 + \left(\frac{w_{pre}}{2}\right)^2}, \tag{3.4}$$

$$\theta_{1in} = \arctan\left(\frac{w_{pre}/2}{H_{Ypre} - l_{ankle} - h_{pre}}\right), \tag{3.5}$$

$$\chi_f = l_1, \tag{3.6}$$

$$\theta_{1f} = \arccos\left(\frac{w_{cur}/2}{\chi_f}\right), \tag{3.7}$$

$$\theta_{3f} = \arcsin\left(\frac{w_{cur}/2}{\chi_{tmp}}\right) + \arccos\left(\frac{l_3^2 + \chi_{tmp}^2 - l_4^2}{2l_3\chi_{tmp}}\right), \tag{3.8}$$

$$\theta_{4f} = \theta_{3f} + \arccos\left(\frac{l_4^2 + \chi_{tmp}^2 - l_3^2}{2l_4\chi_{tmp}}\right). \tag{3.9}$$

ただし，

$$H_{Ypre} = \sqrt{l_1^2 - \left(\frac{w_{pre}}{2}\right)^2} + l_{ankle}, \qquad H_{Ycur} = \sqrt{l_1^2 - \left(\frac{w_{cur}}{2}\right)^2} + l_{ankle}, \tag{3.10}$$

$$\chi_{tmp} = \sqrt{(H_{Ycur} - l_{ankle} - h_{cur})^2 + \left(\frac{w_{cur}}{2}\right)^2}, \tag{3.11}$$

となる.

遊脚に対する初期条件 $(\theta_{3in}, \theta_{4in})$ は動作生成時に，立脚相に対する計算の結果得られる．θ_2 に関しては，初期条件・終期条件共に h/w の値に適当な係数を乗ずることによって決定するが，インプリメント上は通常はゼロに設定してある．また，遊脚足首の軌道に関しては次節で詳しく説明する.

ステップの各フェーズ（立脚相，遊脚相，両脚支持）の持続時間は，第 3.3.1 節の (3.2) 式から計算できるが，ステップの持続時間 T の値は速度（v）と歩幅（w）から容易に得られるので，T^{ds} の値を求めれば全ての持続時間が得られることになる．平地歩行に関しては，T^{ds} の値を求める経験的な公式が存在するが [19]，階段のような狭い歩幅の場合などには適応できないなど制約が大きいため，T^{ds} の値を $T^{ds} = 0.25\ T$ と近似し，広範な歩行動作に対応させている [98]．この係数はもちろん可変である.

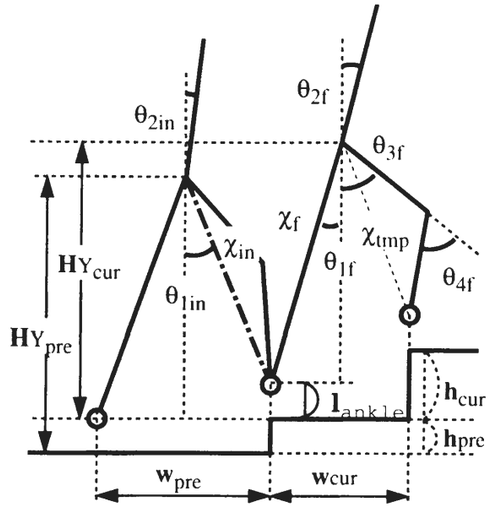


図 3.11: 直進歩行時の境界条件.

さらに，動力学計算後の運動学計算に対する，運動学を適用する部分の可動範囲を求めることが必要である．これは，文献 [86] などを基に，ステップ・オブジェクトの幾何的なパラメータ等から計算する [101].

3.4.5 歩行動作の生成

動作生成は，動力学的手法による生成と運動学的な補間に分けられる．動力学による生成では，動力学的変数（図 3.7）のフレーム時間単位の時間変化を，Lagrange の運動方程式の数値積分（Euler 法）により計算する．運動学による補間では，動力学によって計算された変数を補正するとともに人体の各部分の動きを補い，最終的に 1/60 秒または 1/30 秒ごとの時間変化値として全ての関節角度，および重心の位置を求める.

動力学の計算

動力学的計算の原理は以下のようなものである。
(1) 全身を大まかに 2 つの部分（立脚・上体系と，遊脚系）に分離する．遊脚質量は全身に比べて小さいので，分離しても系全体に及ぼす影響は小さいものとする.

(2) 各パーツは全て鉛直平面上にあるものとし，Lagrange の運動方程式に従う．

この系に対する Lagrange の運動方程式は 2 次の非線形方程式であり，以下の式で表される．

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q_r}}) - \frac{\partial L}{\partial q_r} = F_{q_r}, \quad r = \theta_1, \theta_2, \theta_3, \theta_4, \chi. \tag{3.12}$$

ここで L は系のラグランジアンであり，F_{q_r} は一般化力である．q_r は一般化座標であり，図 3.7の各変数を表す．この運動方程式を用いると，立脚相および遊脚相に対する運動方程式は以下のようになる．ただし，l_i，m_i，I_i はそれぞれ対応するパーツの長さ・質量・慣性モーメントを表し，r_i はパーツ端から重心までの距離を表す：

立脚相：

$$\begin{aligned} F_\chi &= m_2\ddot{\chi} - m_2\chi\dot{\theta_1}^2 - m_2 r_2 \ddot{\theta_2}^2 \sin(\theta_2 - \theta_1) - m_2 r_2 \dot{\theta_2}^2 \cos(\theta_2 - \theta_1) \\ &\quad + m_2 g cos\theta_1, \end{aligned} \tag{3.13}$$

$$\begin{aligned} F_{\theta_1} &= (I_1 + m_1 r_1^2 + m_2\chi^2)\ddot{\theta_1} + 2m_2\chi\dot{\chi}\dot{\theta_1} \\ &\quad - (m_1 r_1 + m_2\chi) g \sin\theta_1 + m_2 r_2 \chi \ddot{\theta_2} \cos(\theta_2 - \theta_1) \\ &\quad - m_2 r_2 \chi \dot{\theta_2}^2 \sin(\theta_2 - \theta_1), \end{aligned} \tag{3.14}$$

$$\begin{aligned} F_{\theta_2} &= -m_2 r_2 \ddot{\chi} \sin(\theta_2 - \theta_1) + m_2 r_2 \chi \ddot{\theta_2} \cos(\theta_2 - \theta_1) \\ &\quad + (I_2 + m_2 r_2^2)\ddot{\theta_2} - m_2 g r_2 sin\theta_2 + 2m_2 r_2 \dot{\chi}\dot{\theta_1} \cos(\theta_2 - \theta_1) \\ &\quad + m_2 r_2 \chi \dot{\theta_1}^2 \sin(\theta_2 - \theta_1). \end{aligned} \tag{3.15}$$

遊脚相：

$$\begin{aligned} F_{\theta_3} &= (I_3 + m_3 r_3^2 + m_4 l_3^2 + I_4 + m_4 r_4^2 + 2m_4 l_3 r_4 \cos\theta_4)\ddot{\theta_3} \\ &\quad + (I_4 + m_4 r_4^2 + m_4 l_3 r_4 \cos\theta_4)\ddot{\theta_4} - (m_3 r_3 + m_4 l_3)(\ddot{x_h}\cos\theta_3 - \ddot{y_h}\sin\theta_3) \\ &\quad - m_4 r_4 (\ddot{x_h}\cos(\theta_3 + \theta_4) - \ddot{y_h}\sin(\theta_3 + \theta_4)) \\ &\quad - m_4 l_3 r_4 \dot{\theta_4}(2\dot{\theta_3} + \dot{\theta_4})\sin\theta_4 + m_3 g r_3 \sin\theta_3 \\ &\quad + m_4 g(l_3 \sin\theta_3 + r_4 \sin(\theta_3 + \theta_4)), \end{aligned} \tag{3.16}$$

$$\begin{aligned} F_{\theta_4} &= (I_4 + m_4 r_4^2 + m_4 l_3 r_4 \cos\theta_4)\ddot{\theta_3} + (I_4 + m_4 r_4^2)\ddot{\theta_4} \\ &\quad + m_4 l_3 r_4 \dot{\theta_3}^2 + m_4 g r_4 \sin(\theta_3 + \theta_4) \\ &\quad - m_4 r_4 (\ddot{x_h}\cos(\theta_3 + \theta_4) - \ddot{y_h}\sin(\theta_3 + \theta_4)). \end{aligned} \tag{3.17}$$

ここで $(\ddot{x_h}, \ddot{y_h})$ は腰の位置の加速度であり，次のようにして求められる．なお，立脚の足首位置 (x, y) は動力学的に一定であると仮定している：

$$x_h = x + \chi sin\theta_1, \quad y_h = y + \chi cos\theta_1, \tag{3.18}$$

であるから，

$$\ddot{x_h} = \ddot{\chi} sin\theta_1 + 2\dot{\chi}\dot{\theta_1} cos\theta_1 + \chi(\ddot{\theta_1} cos\theta_1 - \dot{\theta_1}^2 sin\theta_1), \tag{3.19}$$

$$\ddot{y_h} = \ddot{\chi} cos\theta_1 - 2\dot{\chi}\dot{\theta_1} sin\theta_1 - \chi(\ddot{\theta_1} sin\theta_1 - \dot{\theta_1}^2 cos\theta_1), \tag{3.20}$$

である．

さて，右辺に現れる一般化力はトルクあるいは力を示す非線形項であり，解析的に求めるのはいわゆる逆問題となり困難である．従って，トルクあるいは力の関数形を既知とし，非線形項を

$$F_{q_r} = a f_r(q, t), \tag{3.21}$$

の形に近似し，ステップの終了時刻に関節角度が終期条件と一致するように反復計算することでスカラー項 a を求める [19]．トルク項の関数形 $f(q,t)$ については，古くから人間工学や医学的見地から様々な測定がなされている [86]．KLAW システムでは，これらの測定結果を用いて，F_χ をバネ力で，F_{θ_1} を

$$F_{\theta_1} = \begin{cases} F_1, & 0 \le t \le 0.4\ T^{step}, \\ 0, & 0.4\ T^{step} \le t \le T^{step}, \end{cases} \tag{3.22}$$

となる階段状関数で近似しており，F_{θ_3} は e^{-at} の形を持つ減衰力で表している．また，F_{θ_2}，F_{θ_4} については，動力学計算を施していない [19]．

本論文では，KLAW システムと同様の力あるいはトルク関数を，平地ならびに凹凸地形にも用いている．ただし，本論文においては，歩高がゼロの場合（ステップが平地歩行）のみ χ を動力学的に扱い，それ以外では χ_{in} から χ_f まで線形に変化するものとした．これは，χ に加わる力と歩高との因果関係が明らかでないことや，KLAW システムと同様の計算を行うと，歩幅が小さいとき（例えば階段など）に数値計算上の不安定さが生じることによる．

しかしながら，このような近似を行っても，境界条件を満たすこと以外は，どのような計算結果になるかは知ることができず，例えば望ましい動きを生成するには困難が生じることになる．本論文では，このような問題に対し，動きを生成する力の項として床反力を用いることでより有用な結果を得ており，これに関しては第 4 章で述べることにする．

遊脚の計算

遊脚は全身に対する質量比が小さい（約 16%）ので，運動方程式を分離することによる影響は小さいと仮定している．運動方程式からの計算では，足首の位置は境界での値しか決まっていないため，各時刻における可能な (θ_3, θ_4) の組合せが無数に存在することになる，という典型的な逆問題が生じてしまう．従って，遊脚相における足首の軌道に対し，何らかの拘束式をあらかじめ決定しておき，結果的に股間接角度のみを動力学で求め，膝関節角度は内挿によって求める手法を取る [19]．

足首軌道曲線に対しては，KLAW システムは 4 次曲線を用いているが，本節では，

- 軌道の容易な制御，

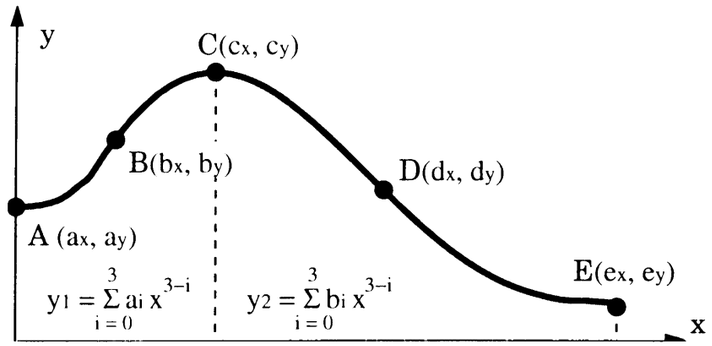


図 3.12: 遊脚足首の軌道.

- 曲線の極大値近傍における数値的不安定性の解消,
- 地形の高低差に応じた各種曲線パターンへの適応,

を実現するために，2 本の 3 次曲線の組合せを用いている．即ち，図 3.12における ABC を 1 つの 3 次曲線，CDE をもう 1 つの 3 次曲線で近似している．2 つの曲線は点 C でなめらかに連続する．これらの曲線を決定するために，図 3.12における点 B，C，D の座標値を地形情報などに応じて，境界条件計算時に計算する（A と E は計算過程で決定される）．

これらの条件を基に，遊脚関節角度の計算の処理手順は以下のようになる．

(1) 遊脚の足首位置は 2 つの 3 次曲線を爪先離地時から踵着地時までなめらかに接続した曲線上を移動する,
(2) 大腿部角度 θ_3 は運動方程式から動力学的に計算する,
(3) 膝角度 θ_4 は図 3.7 のように内挿する,
(4) 足首と爪先の角度は，爪先離地時の角度から踵着地時までの角度までを補間する.

なお，θ_3 の計算に用いるトルク項 F_{θ_3} は，KLAW システムと同様に e^{-at} の形を持つ減衰力で表している．遊脚足部の地面への衝突は幾何的に判定され，遊脚軌道曲線の係数決定へフィードバックされる．

数値積分は，Δt は時間間隔として，以下のような Euler 法を用いて行う：

$$\dot{q_r}(t+\Delta t) = \dot{q_r}(t) + \Delta t\ \ddot{q_r}(t), \tag{3.23}$$

$$q_r(t+\Delta t) = q_r(t) + \Delta t\ \dot{q_r}(t+\Delta t). \tag{3.24}$$

ここで，$\ddot{q_r}(t)$ は，運動方程式を移項して数値的に近似計算を行なうことにより求める．

運動学による補間

運動学による補間は，動力学的計算の後に実行される．この段階では，上記の動力学的計算から得られた各動力学変数のシーケンスを全て適切な関節角度として補正し，運動方程式で表示されなかった上半身などの動きを生成する．主な補間要素は

- 足部の動き,
- 骨盤の（背骨に対する）回転や上下運動,
- 肩の回転,
- 腕の運動,

などであり，処理の流れは以下のようになっている：

(1) 腰の位置を調整する,
(2) 立脚の長さに応じて，股間接角度・膝関節角度・足首関節角度を調節し，大腿部・下腿部を挿入する,
(3) 立脚・遊脚の足部の動きを計算する,
(4) 骨盤，肩，上体，腕などの動きを計算する.

骨盤や肩，腕の動きなどは，動力学で計算された立脚股関節角度の一次関数で表し，連動させている．ただし，本章の後の方では，動力学を用いる手法について言及する．その他の関節角度は最大値と最小値の間を線形的に，あるいは正弦関数を用いて補間している．これらは歩行属性として可動範囲のデフォルト値が設定されているが（例えば肘の最大屈曲角度や肩の回転率など），値を変更することにより種々の特徴を持った歩き方が可能となる．

特に立脚足部は接地の際に踵着地・足裏全体の着地・踵の上昇といった一連の動きを地面から浮き上がったりめり込んだりせずに実行できるよう，常に腰の高さと地面の高さとを比較しながら足首や足裏の位置，関節角度補間している．なお，接地地面が傾いている際にはその近傍の傾きから接地足首角度を調整し，不自然に浮き上がったりめり込んだりしないよう調整している．

図 3.13 は立脚に対する内挿を示している．図 3.13 (a) は踵着地の瞬間である，この図では説明を簡単にするために膝の角度はゼロになっているが，階段等では当然膝が屈曲している．図 3.13

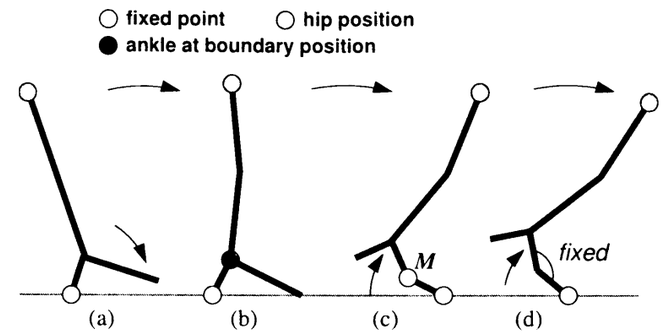


図 3.13: 立脚足部角度の計算; (a) 踵着地, (b) 爪先着地, (c) 他の足の踵着地, (d) 両足支持.

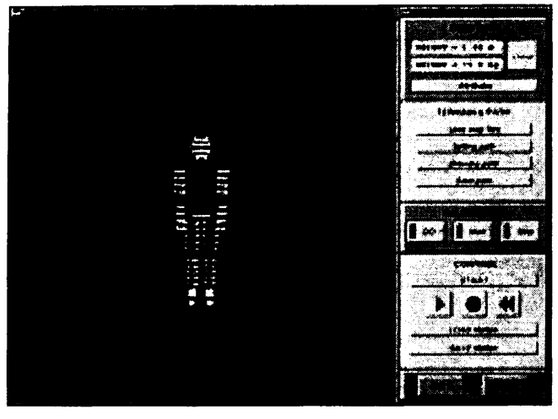


図 3.14: 起動後の GUI 画面.

(b) は重心が足首位置の真上に来た瞬間である．図 3.13 (a) から (b) まで，立脚はと爪先は固定された踵の周りを一定の角速度で回転するものと仮定し，足部と爪先部がなす角度は固定されている．踵の位置は，図 3.13 (b) に示す足首の位置，即ち，ステップの境界 (図 3.11を見よ) における位置から計算することができる．階段歩行においては，我々は爪先と踵が同時に着地するものと仮定し，足首と爪先の位置は (b) まで固定されているものと仮定した．

図 3.13 (b) から (c) までの間，立脚の足部は固定された点 $\boldsymbol{M}$ の周りを，あらかじめ設定された一定の角速度で増加する．最後に，図 3.13 (c) から (d) の間，足と爪先が固定された爪先の周りを回転し，この間，爪先位置と，足中部の角度は固定されており，回転の角速度はやはりあらかじめ計算されている．

全ての期間において，重心位置，足首の位置，足中部角度 は既知であり，大腿部，下腿部ともに長さが一定であるので，実際の大腿部角度，膝角度および足首角度を計算することができる．

3.4.6 表示

以上により出力された各関節角度に基づき，3 次元歩行動作シーケンスを地形形状とともに表示する．このとき，歩行動作の生成は立脚足首を原点としていたが，人物像モデルは左右の部位が面対象となっているため，立脚系＋遊脚系で表されている各数値を，左側＋右側で表されるような数値に変換する必要がある．

出力された関節角度の時系列データは，それ自身が独立したファイルとして保存することも可能であり，より精密な人物像を用いたレンダリング手法で描画することも可能である．付録 A.3 及び付録 B.3 に，C / C ++ 言語または Java 言語で表示するための疑似コードを示す．

(a) (b)

図 3.15: 3 次元地形例; (a) 上部図, (b) 俯瞰図.

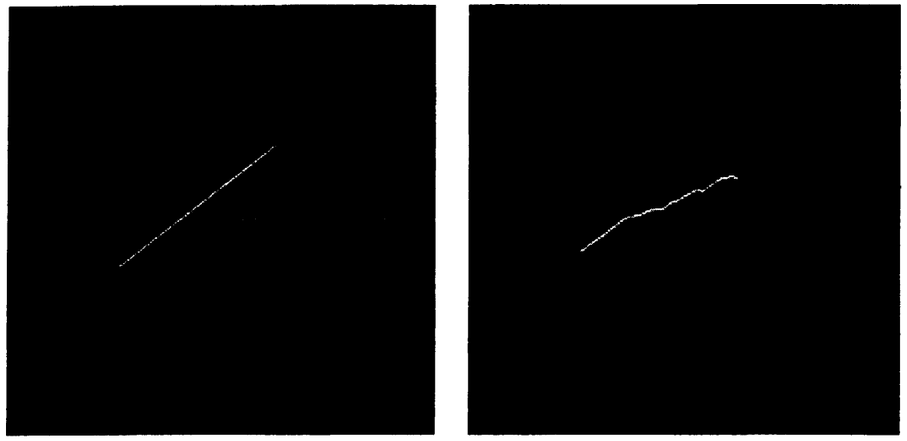
3.4.7 生成結果例

本節で述べた手法を適用した生成例を以下に示す．

本手法を実装したシステムの GUI （Graphical User Interface）画面を図 3.14に示す．左側のレンダリング・ウインドウは計算結果の表示を行い，右側は各種操作やパラメータの制御を行うコントロール・パネルである．コントロール・パネルは “BODY DATA”, “TERRAIN & PATH”, “CALCULATION”, “CONTROL” の 4 つのボックスにより構成され，ユーザはこれらを用いて各種制御を行うことができる．

本手法を実装したシステムを起動すると，まず人物像のモデルが現れる（図 3.14）．“BODY DATA” ボックスを用いて，身長や体重などのモデルの設定値を変更することができる（後述の図 3.34も参照のこと）．次に “TERRAIN & PATH” ボックスを用いて地形データファイルを指定すると，指定された視点から見た地形形状イメージが，レンダリング・ウインドウに描画される（図 3.15）ので，ここでユーザは歩行パスの設定を行う．図 3.16 に示すように，始点と終点をマウスクリックで示せば地形に沿ってパスが自動的に設定される．この段階で，3.4.3節で述べたような “エリア分割” と “ステップ分割” がパスに沿って実行され，歩行シーケンスがアニメーションとして表示される（図 3.17）．

フラクタルを用いて生成した地形形状に，生成された人物像の歩行を加えたシーケンスを図 3.18 に示す．歩行パスは図 3.16 (a) と同様である．図 3.18 から，地形に沿って張られたパス上の歩行を容易に確認することができる．なお，このアニメーションは，人物像モデルを身長 180.0 cm，体重 75.0 kg とし，歩行パラメータを速度 70 m/ 分，歩幅 70 cm として生成したものである．



(a) (b)

図 3.16: パスの設定と表示; (a) 上部図, (b) 俯瞰図.

3.4.8 検討

本節で述べた手法の利点として以下のような事例を挙げることができる：

(1) ユーザは活動的で説得力あるアニメーション映像を得ることができる．地形形状映像等に歩く人物像を加えることにより，シーンに現実味が増す．

(2) 生成した映像から，スケールや表面状態といった地形の特徴を容易に把握することができる．人物像は環境のリファレンスとして重要な役割を果たすため，特に建築物等の映像が理解し易くなる．

(3) 歩行シーケンスを自動的に生成するため，地形データを入力し，歩行パスを設定するだけでアニメーションが生成できる．これはアニメータの負担を軽減する．

(4) 各フレーム（1/60 秒または 1/30 秒）ごとの関節角度の時間変化をファイルとして保存できるため，他のレンダリングシステム等，より複雑なアプリケーションとの結合が可能である．

動作の生成は，運動方程式の計算における時分割を 1/60 秒としているため，本論文執筆時においては実時間の計算が可能である．しかしながら，1 歩の動作生成を，全て踵着地の瞬間に行っているため，踵着地の瞬間に，数フレーム程度の遅延が見られるという問題がある．これは，歩行途中で地形や移動速度等が変化した場合でも，修正が 1 歩ごとに可能となるように拡張可能とするためであるが，不連続な印象を与えることがある．リアルタイム性に関しては，計算機の性能に依存するところがあるため，正確な評価は難しいが，第 3.6 節で計算時間について検討することとする．

本節で述べた手法をさらに発展させ，より有用な応用を行うために，

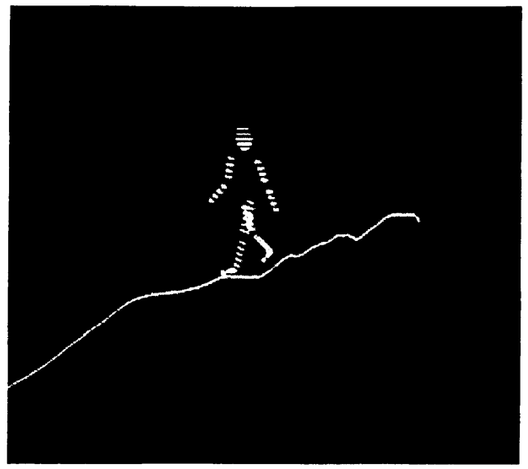


図 3.17: パスに沿った歩行動作の生成例.

- 運動方程式を採用するためのモデル化を，より効率的なものとする，
- 左右への方向転換などを可能とする，
- 方程式における一般化力の項を検討する，
- 複数人の生成を可能とする，

などの検討を行う必要があり，これらについては本節以降で述べていく．なお，本研究においては，あらかじめ歩行するパスを設定しているため，自律的な障害物回避などには対応していないが，この問題に関しては第 5 章で言及する．

3.5 腕の動作の生成

腕の運動は能動的であるか受動的であるかにかかわらず，本質的には体幹の回旋を抑制したり，逆に歩行動作を大きくするための動きである．上腕角度に関しては，通常の歩行では平均的に前方に約 30 度，後方に約 9 度の振りがあることが知られている [68].

その際，例えば右腕に着目すると，図 3.3に示すように，各時刻における境界条件は右踵着地の瞬間（図 3.3 の Heel strike (R)）に後方への最大伸展となり，左足離地の瞬間（同 Toe off (L)）にちょうど体側，左足着地の瞬間（同 Heel strike (L)）に前方への最大屈曲となる．さらに右足離地の瞬間（同 Toe off (R)）に体側，再び右踵着地の瞬間（同 Heel strike (R)）に後方への最大伸展となる．従って，運動学的見解からはこの境界条件間を適当な方法で補間すれば腕の運動を表現できることになるが，動作が単調になりがちである．

逆に動力学的計算の場合は，地形への対応を考慮しなくてもよいため，足部ほどの厳密性は要求

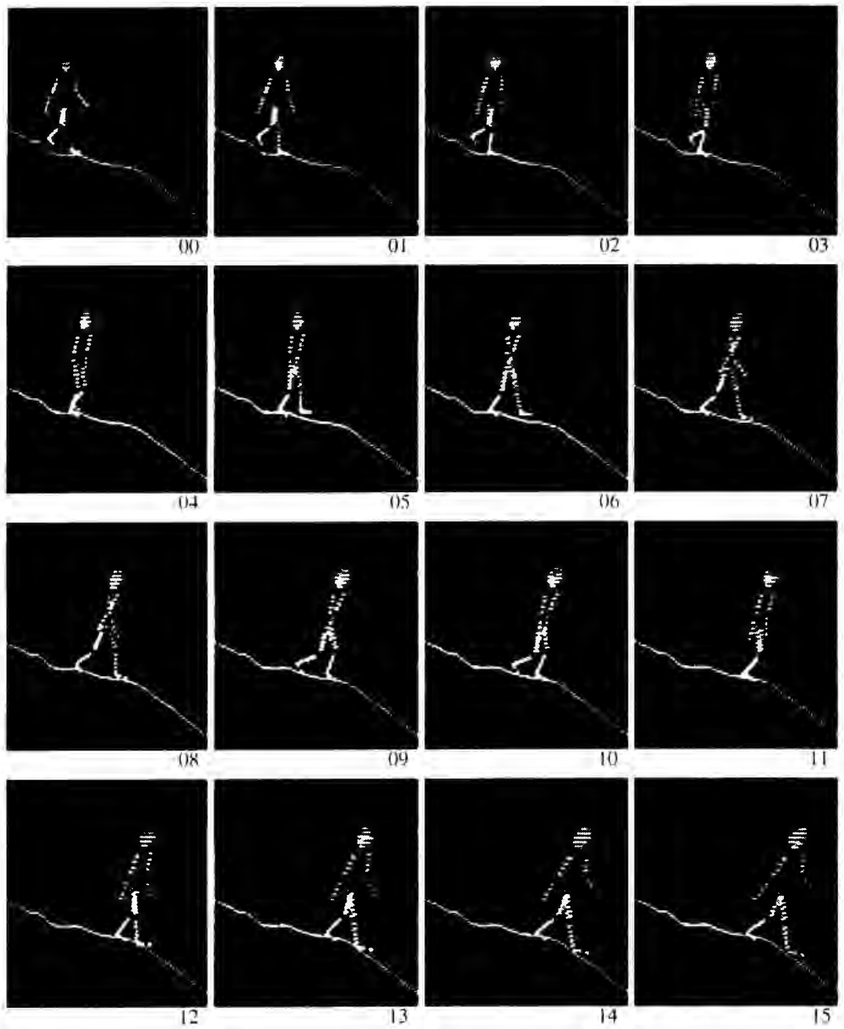


図 3.18: 歩行シーケンス生成例.

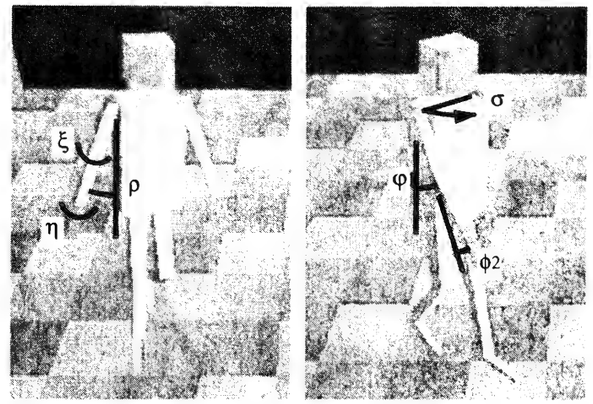


図 3.19: 腕の変数; (a) 上腕・下腕の回旋と脇の角度, (b) 肩，前後への角度，肘角度.

されないが，それでも境界条件を考慮して関節に加わるトルクを計算する必要があり，複雑なモデル化では計算コストがかかるという問題がある.

歩行動作に限定されない，腕の運動そのものに対する能動的な Goal-oriented なモデリング手法には Koga らの経験的な拘束式を用いた手法 [53] や Lee らの軌道による拘束を用いた手法 [56] などがあるが，逆問題に対する定式化や計算時間などの面で困難が伴う．また，歩行動作に伴う従属的な腕の運動に関する報告は Boulic らの補間を主体とする手法 [14] や Unuma らのやモーション・キャプチャーをベースとする手法 [109] などの運動学的手法が多く，動きが単調になることや入力の手間がかかること，動作のデータベースを用意しなければならないという問題点があった. また，自動生成においても，モデリングの過程における動力学計算の定式化において腕の運動は考慮しておらず，大腿部の運動に従属する形で運動学的に生成していた例が多い [19][43].

これに対し，腕を二重振子で近似して動力学により計算した結果と，運動学的な補間による結果とを重ね合わせる手法 [100] なども見られるが，計算上の不安定性などが生じることがあった.

このため，本節では，腕全体の動きを動力学により計算し，その後，肘の運動を運動学的に補間する手法を採用する [107]. この手法は，何らかの加重が腕にかかっている場合，即ち手に質量のある物体を持っているような場合でも対応可能である.

3.5.1 腕の運動に対する運動方程式

図 3.19 は，腕の運動に置ける変数を示している．図 3.19(a) において，ξ は上腕部の回転，η は下腕部の回転であり，ρ は脇角度である．図 3.19(b) において，σ は肩の回転角度，φ_1 は上腕部の前後への角度，φ_2 は肘角度である.

本節で述べる手法においては，ξ，η は考慮しない．また，他の角度も直接に動力学計算を行わ

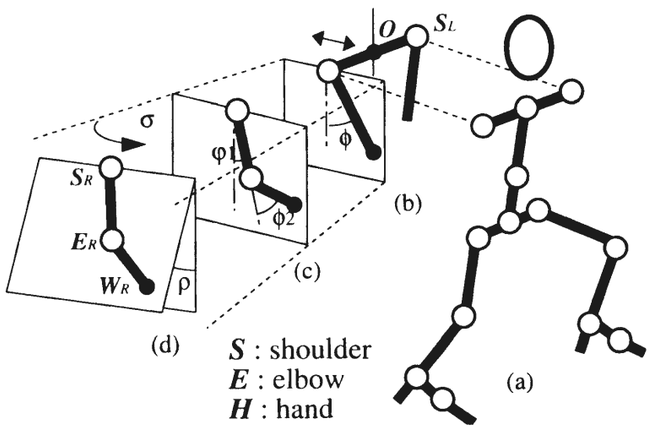


図 3.20: 腕のモデル; (a) 人体モデル, (b) 前後への運動, (c) 肘の運動, (d) 腕の広がり.

ず，腕全体を 1 つの剛体棒で近似したものに動力学計算を適用し，φ_1 および φ_2 を求め，σ は骨盤の運動から計算し，ρ は角度を直接指定する．

図 3.20 は本節における腕の運動のモデルを示している．

このモデルおける計算手順は以下の通りである：

(1) 腕全体を近似した剛体棒の角度 ϕ を動力学的に計算し，上腕角度 φ_1 を計算する，

(2) 肘角度 φ_2 を運動学的に計算する，

(3) 腕モデルが背骨の周りに肩を介して σ で回転し，角度 ρ で脇の下が持ちあがる．

まず，二重振子の計算は不安定であるため，腕全体を近似した剛体棒の角度 ϕ が計算される．この剛体棒の運動モデルをを図 3.21に示す．

l_1，l_2 をそれぞれ上腕と下腕の長さ，m_1，m_2 を質量とし，m を手 (あるいは荷重) の質量とする．$\boldsymbol{S}(X,Y,Z)$ を肩の位置とし，$\boldsymbol{G}(x,y)$ を系の重心とする．

図 3.21 に示すように $\boldsymbol{S}(X,Y,Z)$ は鉛直面内を運動するものと仮定し，腕を，支点が $\boldsymbol{S}$ で質量が $M_a = m_1 + m_2 + m$ である物理振子で近似する．さらに，図 3.22 に示すように，ϕ が計算されている間，肘角度 β_c は一定であると仮定する．以上によって，$Y = const.$，$Z = 0$，$\phi = \varphi_1 + \gamma$ とみなすことができる．γ は $\overline{SG}$ と $\overline{SE}$ の間の角度であり，

$$\cos\gamma = \frac{1}{2RM_a}\,[\, l_1(m_1 + 2m_2 + 2m) + l_2(m_2 + 2m)\cos\beta_c \,], \tag{3.25}$$

で表される．

このモデルにおいて，ϕ に対する運動方程式は以下のようになる：

$$I_a\ddot{\phi} + M_aR\,(\,\ddot{X}\cos\phi + g\sin\phi\,) = F_a, \tag{3.26}$$

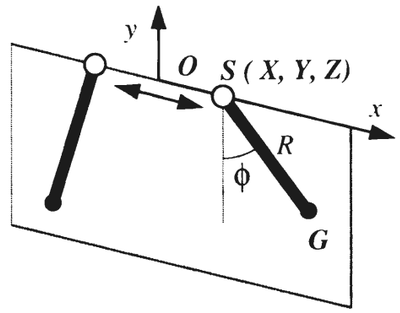


図 3.21: 腕全体の運動モデル

ここで，I_a は腕全体の慣性モーメントであり，その軸は $\boldsymbol{S}$ を通り，xy 平面に垂直である．

$$I_a = (\frac{m_1}{3} + m_2 + m)\; l_1^2 + (\frac{m_2}{3} + m)\; l_2^2 + (m_2 + 2m)\; l_1 l_2 \cos\beta_c, \tag{3.27}$$

R は系の回転半径であり，以下を満たす：

$$\begin{aligned} R^2 \;=\; & \frac{1}{4M^2}\,[\; l_1^2(m_1 + 2m_2 + 2m)^2 + 2l_1l_2(m_1 + 2m_2 + 2m)(m_2 + 2m)\cos\beta_c \\ & +\; l_2^2(m_2 + 2m)^2\;]. \end{aligned} \tag{3.28}$$

肩の位置 S は肩の回転角度 σ と，肩の長さを用いて計算される．肩の速度 $\dot{\boldsymbol{S}}$ とその加速度 $\ddot{\boldsymbol{S}}$ は，以下の回転運動から計算可能である：

$$\sigma(t) \;=\; -factor \cdot pelvic\_rotation(t), \tag{3.29}$$

$$X(t) \;=\; harf\_length\_of\_shoulder \cdot \sin\sigma(t). \tag{3.30}$$

角 $pelvic\_rotation(t)$ は，直進歩行動作の計算において運動学的に計算される骨盤の鉛直線周りの回転角度であり，本研究では，正弦関数を用いた補間で運動学的に計算される．また，$factor$ は適当な因子である．$harf\_length\_of\_shoulder$ は，両肩を結ぶ距離の半分の長さを示す．

(3.27) 式における運動を簡単にするために，本節においては，トルク F_a をステップ関数で近似した [107].

$$F_a(t) = \begin{cases} f_a^1, & 0 \le t \le T^{ds}, \\ f_a^2, & T^{ds} \le t \le T^0, \end{cases} \tag{3.31}$$

詳しくは第 4 章で説明するが，T_0 は前後方向の床反力の値が 0 になる時刻である．f_a^1 は加速のトルクであり，f_a^2 は減速のトルクであり，これらの値は，以下の境界条件を満足するようにこれらの値を変更しつつ (3.26) 式を反復して計算することによって求める：

$$\dot{\phi}(0) = \dot{\phi}(T) = 0, \tag{3.32}$$

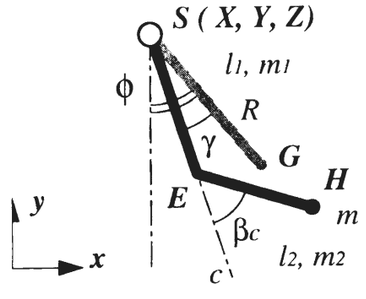


図 3.22: 腕の幾何的近似モデル

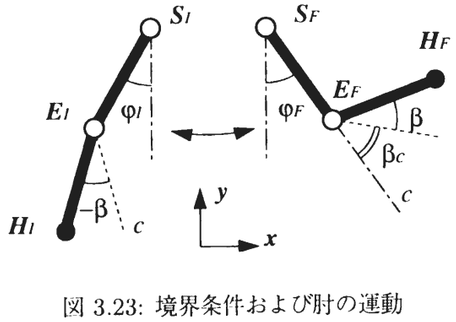


図 3.23: 境界条件および肘の運動

かつ

$$\varphi_1(0) = \varphi_I,\ \ \varphi_1(T^{ds}) = 0,\ \ \varphi_1(T) = \varphi_F, \tag{3.33}$$

$$\varphi_1(0) = \varphi_F,\ \ \varphi_1(T^{ds}) = 0,\ \ \varphi_1(T) = \varphi_I. \tag{3.34}$$

(3.33) 式の方程式は前方へのスイングであり，(3.34) 式は後方へのスイングである．この手法では，(3.32) 式と $t = 0$ や $t = T$ での角度や角速度の連続性は厳密に考慮されるが，(3.33) 式や (3.34) 式はそれほど厳密には適用されない．

3.5.2 肘角度の計算

次に，肘角度を φ_1 を用いて計算する．肘角度は後方に曲がることはないので，この角度は $\varphi_2 \geq 0$ を満たさなければならない．従って，図 3.23 に示すように，肘の角速度は，上腕からの延長線から β_c をなす線の周りの正弦関数で近似する．

最後に，肩の回転角 σ と脇の拡がりを表す角度 ρ が加えられる．σ は前述の通り，骨盤の回転角度に関連づけて計算し，ρ はユーザによって直接設定される．

これらの定式化において，腕の運動は身体の他の部分の運動に影響を与えないと仮定している．

3.6 部分運動への分離と再構成

本研究の目的である 3 次元仮想空間内に設定された任意のパスに沿った歩行動作を生成するためには，第 3.4 節で述べた直進歩行のみならず，左右への方向転換に対応する運動を生成することが必要である．本節では，左右への方向転換を含む歩行動作の生成手法について述べる．

第 2 章でも述べた通り，左右への方向転換は，主に foot print アプローチに基づいて，あらかじめ次のステップの着地点を指定しておき，現在の着地点と指定された次の着地点間を補間する運動生成法が主流である．例えば，Girard [38] や Overveld ら [73] は，足の位置をあらかじめ決定し，運動学的に回転させることで方向転換を実現しており，Bezault ら [13] は歩行の軌跡を対話的に扱う手法を提案し，van de Panne [75] や Torkos [95] は動力学的な手法を用いた足跡間の移動を実現している．

本節で述べる手法は，3 次元仮想空間内に対し，任意の歩行パスを設定するだけで，そのパスに沿う歩行動作を自動生成するものであり，動作生成の入力パラメータの軽減や歩行動作の誇張表現を可能としている．本手法の特徴は，

(1) 1 歩の歩行動作を直進歩行動作部分と回転動作部分に分離し，それぞれの動作生成おいて動力学計算と運動学計算とを融合させて計算する，

(2) 地形情報に合わせて境界値を変更することで，パスの形状に沿った歩行の誇張表現を可能とする，

という点にある [101]．これにより，高低差ある地形への対応に加えて回転動作による方向転換が可能となり，3 次元でモデル化された地形に適応した歩行動作の自動生成を実現し，左右転換を含む歩行動作を，直進歩行動作と回転動作に分離して動力学計算を行い，生成後に再び融合することで，計算量を軽減することを可能とし，実時間での歩行動作生成を実現した．

本節で述べる手法においても，ステップごとの境界において関節角度や重心位置などの速度は保存するが，基本的に 1 歩の歩行動作を独立して計算し，後述するパスのステップへの分割によって計算される境界条件等に応じて，ステップ毎の関節角度を全フレームにわたり計算する．

本手法でも，第 3.4 節で述べた直進歩行動作生成と同様に，

1. 人物像モデルに対する設定，
2. 環境モデルに対するパスの設定，
3. パスのステップへの分割および境界条件の計算，
4. 動力学計算，
5. 運動学計算，
6. 表示，

という手順を踏む．

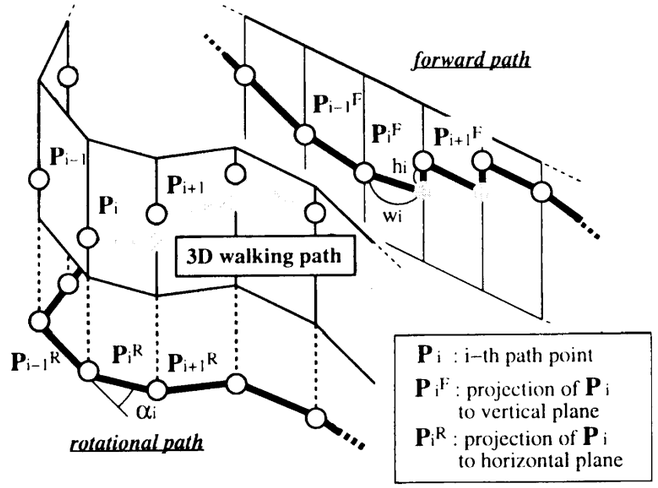


図 3.24: パスの分解.

3.6.1 パスの分解

本節では，直進歩行動作と回転運動を融合させることにより，高低差があり，かつ，曲線的なパスに沿った歩行が可能である．そのために，3 次元仮想環境に設定された歩行パスを，直進歩行動作に対応する部分と回転運動に対応する部分とに分解する [101].

まず，仮想環境に対し，歩行パスを設定する．曲線的なパスにおいて，曲率が大きいときは，適当に補間を行う．本節での生成例では，回転角度が角度にして 90 度を越えないように保管して設定を行う．その後，3 次元パスに対して第 3.4 節で述べた “エリア分割” 及び “ステップ分割” を実行する．本節では主に室内をコンピュータ・グラフィックスでモデル化した仮想環境を想定しており，階段や床などの高低差を利用してエリア分割を行う．具体的には，3 次元パスと仮想環境との交点を求め，その交点の距離や高さによってエリアを生成する．パスの設定方法は，マウス等で線分の両端を次々に指定して行く方法を想定しており，2 つの直線パスの交点は，なめらかに接続するように補間する（後述の図 3.35 を参照のこと）．

また，階段の幅や人物像モデルの標準的な歩幅をもとに “ステップ分割” を行う．その後，分割されたパスを，直進歩行動作部分と回転部分とに分解する．

図 3.24は，ステップ分割後の歩行パスとその分解を示す：ステップ分割後は，“3D walking path” に示すようにステップを表す連結された線分でパスを近似する．

2 つの連続する点が同じ水平位置を有するときは，高さの高い方の点のみが点 P_i として用いられる．直進パス（forward path）は “3D walking path” の各ステップを一つの鉛直面上へ展開したものであり，回転パス（rotational path）は水平面上へ正射影したものである．直進パス上の歩行ス

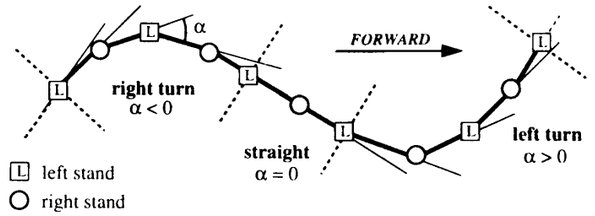


図 3.25: 回転パス.

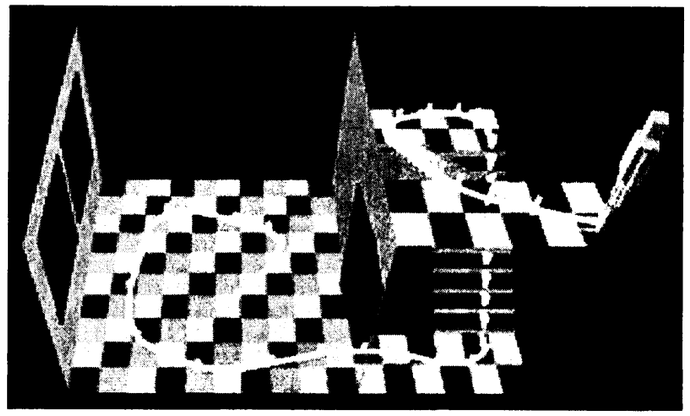


図 3.26: パス設定及びステップへの分割の例.

テップは，それぞれ歩幅（w_i），歩高（h_i）及び角度（α_i）を有し，回転パスは回転角度 α_i を所持する．α_i は，一歩前のステップ $P_{i-1}P_i$ と現在のステップ P_iP_{i+1} との，水平面上における回転角度である．直進歩行動作は，図 3.24における直進パスに沿う動作であり，歩幅 w_i，歩高 h_i を持つステップに関する歩行を生成する．回転動作は，直進パスにおけるステップの持続時間に α_i だけの回転を生ぜしめる動作を生成する．

図 3.25 は回転パスを示す．図 3.25 において，進行方向に対し右に方向転換するか左に方向転換するかによって，α の方向性すなわち正負の符号を決定する．

さらに，各ステップは幾何的な地形情報だけではなく，歩行に関するパラメータ即ち速度（v_i），そのステップの持続時間（T_i），及び左右いずれが立脚であるかという情報（D_i）を持ち，これらの条件は直進歩行動作の生成，および回転運動の生成の両方に用いる．

本手法では直進歩行動作の生成と同様に，あるステップの歩行動作を生成するために直前のステップ及び直後のステップの地形情報が必要であり，また，後述するように連続するステップを滑らかに連結するために，パスを設定すると直ちに全てのステップに関してこれらの情報を計算する．こ

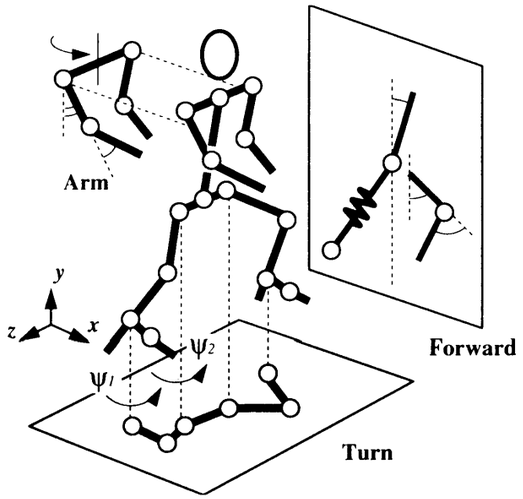


図 3.27: 動力学変数.

れらの情報のうち，パラメータ $[D_i, T_i, w_{i-1}, h_{i-1}, w_i, h_i]$ は直進歩行動作の計算に用いられ，パラメータ $[D_i, T_i, \alpha_i]$ は回転運動に用いられ，そして位置 P_i はローカルな座標原点に取られる．直進歩行動作において，h_i の値や左右のどちらに方向転換するか，に応じて適切な鉛直床反力が適用される．方向は α_i の符号によって決まる．本手法では，このパラメータのみで歩行動作を生成することができる．しかも，歩幅，歩行，角度はパス設定後自動的に計算されるため，ユーザは基本的に移動速度のみを入力すれば良いことになる．

図 3.26 は，実際に室内をモデル化した 3 次元仮想空間にパスを設定し，ステップに分割したシーンの例を表している．図 3.26 から，階段を含む複雑な地形においても，適切にステップへの分割が行われていることが分かる．

3.6.2 動力学変数

本節における人物像の動力学モデルを図 3.27に示す．本手法では，歩行動作の動力学部分を，直進歩行動作（図 3.27の Forward），回転動作（図 3.27の Turn），腕の運動（図 3.27の Arm）に分解する．

このうち，直進歩行動作および腕の運動に関しては，第 3.4 節，第 3.5 節でそれぞれ説明した．いずれも，これらの系に関する Lagrange の 運動方程式

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_r}\right) - \frac{\partial L}{\partial q_r} = F_{q_r}. \tag{3.35}$$

を立て（記号の意味は第 3.4 節と同様である），トルクあるいは力の関数形 $f_r(q,t)$ を既知とし，

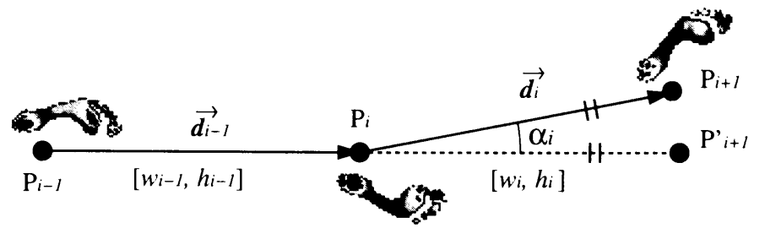


図 3.28: 境界条件.

ステップの終了時に関節角度が終期条件と一致するように反復計算することで，トルクあるいは力を求めつつ，関節角度を計算することになる．

本手法では，直進歩行動作に関する動力学変数，回転動作に関する動力学変数および腕の運動に関する動力学変数は，互いに動力学的に独立であり，動力学計算においては互いに干渉することはないものと仮定した．

実際の歩行動作は，様々な要素が互いに関連しあっている複雑な運動であるが，干渉する部分は動力学計算後に，運動学的に誇張したり修正を行う．なお，後述する生成結果において，この仮定が特に不都合をもたらさないことを確認している．

図 3.28 は，前節で述べたパス分割後の境界条件を示す．図 3.28 において，第 2 章の図 3.4 と同様に，$\boldsymbol{P}_i$ は i 番目の踵着地点を表している．$\boldsymbol{P}'_{i+1}$ は，$\boldsymbol{P}_{i-1}$ と $\boldsymbol{P}_i$ を結ぶ直線の延長線上にあり，$\boldsymbol{P}_i\boldsymbol{P}_{i+1} = \boldsymbol{P}_i\boldsymbol{P}'_{i+1}$ となるような点である．実際は $\boldsymbol{P}_{i-1}\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ という歩行動作を生成するが，直進歩行動作，および腕の運動は $\boldsymbol{P}_{i-1}\boldsymbol{P}_i\boldsymbol{P}'_{i+1}$ という歩行動作を生成し，同じ持続時間で身体の向きを $\boldsymbol{P}_i\boldsymbol{P}'_{i+1}$ から $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ まで，α_i だけ回転することになる．

3.6.3 回転運動

回転運動は立脚足首位置に置いた鉛直上向きに伸びるの軸まわりに立脚足首より上の身体部位及び遊脚を回転させることで生成する．既に述べたように，本節ではこの回転運動を直進歩行と独立なものとみなし，直進歩行動作生成から導出される骨盤や上体の回転などの要素とは独立に，純粋に方向転換に関する回転のみを扱う．

回転に関する動力学変数は，図 3.27 に示す通り，立脚足首まわりの回転角 $\psi_1(t)$ と立脚股関節まわりの回転角 $\psi_2(t)$ である．$\psi_1(t)$ は立脚足首から立脚股関節までの回転（立脚のねじれ）を表し，$\psi_2(t)$ は立脚股関節における上体及び遊脚の回転を表す．

さて，図 3.29に示すように，現在歩行中のステップ i 及び直前のステップ $i-1$ の，水平面内での方向をそれぞれ $\vec{d}_i$，$\vec{d}_{i-1}$ とし，ステップ i 内の時刻 t $(0 \le t \le T_i)$ において人物像の前面が向く方向の水平成分を $\vec{d}$ とする．$\vec{d}_{i-1}$ と $\vec{d}$ とのなす角度を $\Psi(t)$ とおくと，

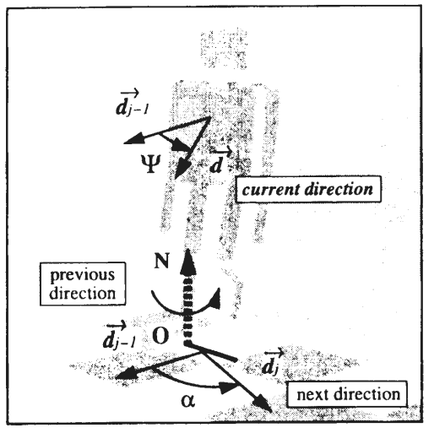


図 3.29: 回転のモデル.

$$\Psi(t) = \psi_1(t) + \psi_2(t), \tag{3.36}$$

と表せる．さらに，通常直進歩行における股関節回転及び足首回転の可動範囲 [86] から得られる比を用いて $\psi_1(t) : \psi_2(t) \simeq 2 : 3$ と近似し，$\Psi(t)$ に関する運動方程式を解くことで $\psi_1(t)$ 及び $\psi_2(t)$ を求める．この近似により，回転の変数は 1 つとなり，式が単純化される．

本手法においては，回転運動を図 3.30に示すような単純な回転モデルで表している．このモデルでは，人体は楕円体で近似されており，その全質量が M，慣性モーメントが $I_r = I_{Gr} + Mr_r^2$ である．ここで I_{Gr} は回転軸が重心を通る慣性モーメントである．回転軸 l_r は立脚足首 O_r を通る鉛直線であり，人体の重心を通る軸は l_r から r_r の距離にある．

外部力 $\boldsymbol{F}_r(t)$ のみがこの系に適用されている場合，$F_r(t)$ を $\boldsymbol{F}_r(t)$ の絶対値とすると，回転の運動方程式は

$$I_r\ddot{\Psi}(t) = r_r F_r(t), \tag{3.37}$$

とる．この手法は r_r は回転の間一定であることを仮定している．実際は歩行に伴い，r_r の大きさは変化するが，運動方程式を簡素に定式化するために一定であるものとした．これは，回転半径方向の動力学的計算を無視することになるが，後述する生成結果例によりこの近似で十分であることが確認されている．

(3.37) 式における歩行中の回転を生じる力 $F_r(t)$ は，実際には体幹の回転を制御する腕の運動や立脚足への前後方向・左右方向への力などが複雑にからみあって生じるが，通常直進歩行では左右方向への力は垂直方向・前後方向に比べ小さいため [86][83]，回転においても前後方向への床反力の影響を最も大きく受けるものと仮定した．床反力に付いては次章で詳しく述べるが，足圧測定から得られる前後方向への床反力を近似した関数形を用いて $F_r(t)$ を表し，$\Phi(T) = \alpha$ となるように

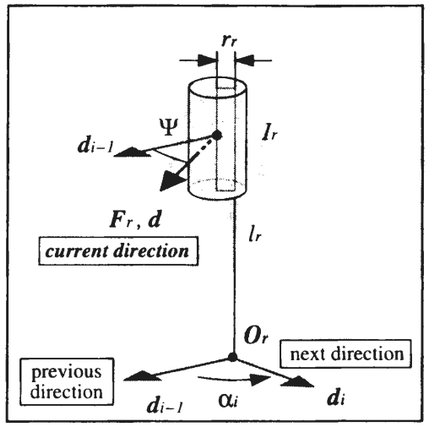


図 3.30: 回転の動力学モデル.

反復計算を行うことによりスカラー因子を求め，各フレーム毎の角度 $\Psi(t)$ を計算する．この $F_r(t)$ は，左右どちらが立脚か，左右どちらへの方向転換かによって符号が異なる．

3.6.4 ステップの連結

各ステップの境界条件に合致するように動力学計算を行った後は，なめらかに連続するステップの列を生成するようにステップの連結を行う．まず，人物像モデルの関節角度を，直進歩行の計算による角度と，腕の運動の計算による角度，そして回転による角度とを加えることによって合成する．このとき，

- ステップの幾何的な境界条件および持続時間を満たすこと，
- ステップの境界で関節の角速度が保存されること，
- 可動範囲が閾値を越えないこと，

を満たすように角度を合成する．その際，各ステップが有する **歩行モード** を考慮しながら計算する．**モード** は歩行動作における，同じ動作が連続する状態を示す．鉛直方向のモードは平面歩行や上昇，下降などであり，水平方向のモードは直進歩行や左右への方向転換である．これらのモードはステップの有する境界条件から決定可能であり，次節で述べる誇張表現を行う際に考慮される因子である．図 3.31 はステップとモードの例を示す．

3.6.5 誇張表現

本手法は，3 次元経路上の人物像の歩行運動に，経路に応じた動作表現や誇張表現を加えることが可能である．これは歩行動作を生成する際に，人物像モデルの関節角度を曲線経路の幾何的形状

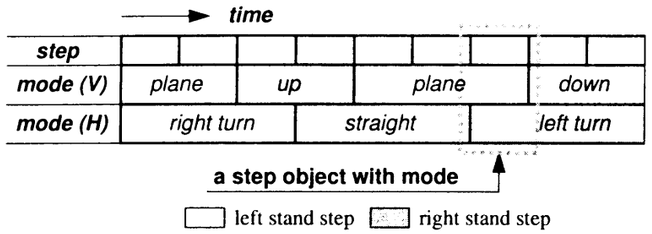


図 3.31: ステップの歩行モード.

を特徴づけるパラメータの関数として表すことにより実現する．具体的には人物像モデルの動力学あるいは運動学的変数に対する上下あるいは左右方向への可動範囲を計算する．これは，

- 動力学計算前に適用し，動力学計算の境界条件とする，
- 動力学計算後に適用し，動力学計算結果の関節角度を修正しつつ合成する，

という 2 つの適用が可能である．

まず，第一の適用例として，上下方向に動く関節角度に対する誇張表現について述べ，次に左右方向に動く関節角度に体する誇張表現に付いて述べる．

上下方向

動作制御レベルで実行したパスの分解に基づき，直進パスの変化に応じて自動的に境界条件を決定する．即ち，図 3.32 (a) に示す上体角度 θ_2，頭部の前後方向への傾きを示す角度 μ 及び上腕角度 φ に対して初期条件及び終期条件を決定し，ローカルな歩行動作生成での動力学的あるいは運動学的計算のための境界条件とする．

具体的な境界条件決定方法の一例を表 3.2 に示す．表において，(0) はデフォルト値をそのまま使うことを，(+)，(−) はそれぞれデフォルト値よりも増加及び減少させることを示し，→ は初期条件から終期条件への変化を示している．また，up，down，plane はそれぞれ地形を表す．

左右方向

左右への方向転換を導入する際には，歩行動作の左右非対称性が生じる．本手法では，第 3.6.4 節で述べたように，回転パス上のあるステップと直前のステップがなす角度 α の符号により水平運動のモードを判断する．即ち $\alpha = 0$ であれば直進歩行モード，$\alpha > 0$ であれば左への方向転換モード，$\alpha < 0$ であれば右への方向転換モードである（図 3.25参照）．これらのモードに応じて，修正を加える変数は図 3.32 (b) で示す体軸の角度 λ，回転内側上腕角度 φ_{int} 及び回転外側上腕角度 φ_{ext} である．

直線歩行の場合，人物像の平均重心位置は歩行経路に平行であるが，本手法においては左右への方向転換をよりリアルに表現したり，あるいはユーザによる誇張表現を可能とするため，曲線経路

表 3.2: 境界条件例

	FROM	TO		
		up	down	plane
θ_2	up	(0) → (0)	(0) → (−)	(0) → (0)
	down	(−) → (0)	(−) → (−)	(−) → (0)
	plane	(0) → (0)	(0) → (+)	(0) → (0)
μ	up	(+) → (+)	(+) → (−)	(0) → (0)
	down	(−) → (+)	(−) → (−)	(−) → (0)
	plane	(0) → (+)	(0) → (−)	(0) → (0)
φ	up	(−) → (−)	(−) → (−)	(−) → (0)
	down	(−) → (−)	(−) → (−)	(−) → (0)
	plane	(0) → (−)	(0) → (−)	(0) → (0)

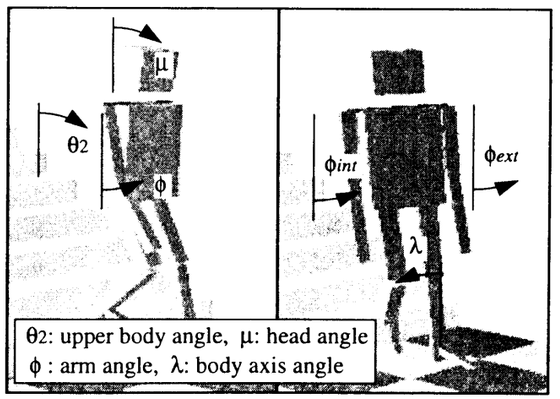


図 3.32: 境界条件の修正例; (a) 上下方向, (b) 左右方向.

に対して人物像の体軸を傾け，平均重心位置が通る曲線を図 3.33で示すように修正する．即ち，あるモードが a 歩めのステップ S_a を起点とする m 歩のステップ $\{S_a, S_{a+1}, \ldots, S_b\}$ $(b = a + m - 1)$ から成るものとし，各ステップ S_i $(i = a, a+1, \ldots, b)$ の開始時刻を τ_i^0，終了時刻を τ_i^s $(= \tau_i^0 + t_i^s)$，τ_i^c を τ_i^0 と τ_i^s の中間時刻として，S_i における j 番目の時刻 t_{ij} での体軸の傾き λ_{ij} を

$$\lambda_{ij} = f\Lambda_{ij} \sin(\pi \frac{t_{ij}}{T}) \ / \ (\sum_{k=a}^{b} |\alpha_k|), \tag{3.38}$$

ただし

$$\Lambda_{ij} = \begin{cases} \Lambda_{i0} + \dfrac{\alpha_i - \alpha_{i-1}}{\tau_i^c - \tau_{i-1}^c} t_{ij}, & (\tau_i^0 < t_{ij} < \tau_i^c), \\ \Lambda_{ic} + \dfrac{\alpha_{i+1} - \alpha_i}{\tau_{i+1}^c - \tau_i^c} t_{ij}, & (\tau_i^c < t_{ij} < \tau_i^s), \end{cases}$$

$$T = \sum_{k=a}^{b} t_k^s, \tag{3.39}$$

とする．ここで f はユーザによって指定される表現因子で，図 3.32 (b) のように，回転方向に体軸を傾けるような通常の歩行のような効果を生ぜしめたい時は $f > 0$ として曲率中心の方へ体軸を傾ける．逆にアニメーション・キャラクタに見られるように遠心力が加わるような効果を生ぜしめたい時は $f < 0$ とする．Λ_{ij} は回転角 α_i の大きさによる体軸の傾きの変化を表現する因子であり，Λ_{i0}，Λ_{ic} はそれぞれ t_i^0 $(= t_{i-1}^s)$，t_i^c での Λ_{ij} の値である．さらにモード境界での値として，$i = a$ の場合の前半部分，及び $i = b$ の場合の後半部分は以下のようにした：

$$\Lambda_{aj} = \frac{\alpha_a}{\tau_a^c - \tau_a^0} t_{aj}, \qquad (\tau_a^0 < t_{aj} < \tau_a^c), \tag{3.40}$$

$$\Lambda_{bj} = \Lambda_{bc} - \frac{\alpha_b}{\tau_b^s - \tau_b^c} t_{bj}, \quad (\tau_b^c < t_{bj} < \tau_b^s). \tag{3.41}$$

φ_{int}，φ_{ext} に関しては，3.6.5節で決定した φ の境界条件をモードに応じて修正する．直進歩行モードの場合はそのままの値を用い，方向転換モードの場合は $|\alpha|$ の値に応じて φ_{int}，φ_{ext} の可動範囲の絶対値 $||\varphi_{int}||$，$||\varphi_{ext}||$ をそれぞれ増減させる．即ち，φ の可動幅を $||\varphi||$ として以下のように表す：

$$\begin{aligned} ||\varphi_{int}|| &= (\ 1 - g_{int}|\alpha|\ )\ ||\varphi||, \\ ||\varphi_{ext}|| &= (\ 1 + g_{ext}|\alpha|\ )\ ||\varphi||. \end{aligned} \tag{3.42}$$

ただし g_{int}，g_{ext} はユーザによって指定される 0 または正の値を持つ表現因子である．

3.6.6 生成例

本手法を 3 次元モデルとして定義された室内シーンに適用した CG アニメーション生成例を以下に示す．本節では，ハードウエアはシリコングラフィックス社 Indigo2^{TM} HIGH IMPACT^{TM} (R4400, 250MHz) を用い，表示には OpenGL^{TM} を用いている．

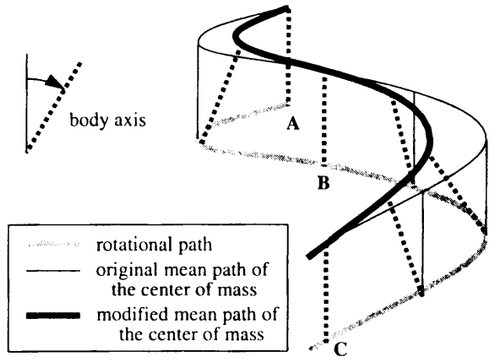


図 3.33: 回転に対する体軸の変化例

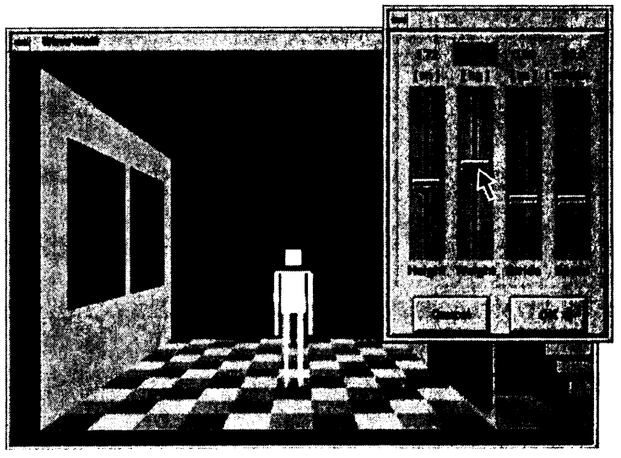
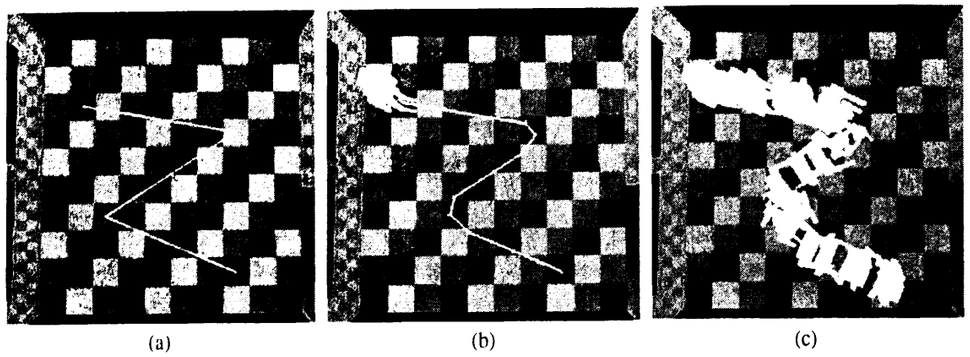


図 3.34: 人物像モデル設定の GUI.



(a) (b) (c)

図 3.35: 曲線歩行例; (a) パスの設定, (b) パスのステップへの分割, (c) 動作生成.

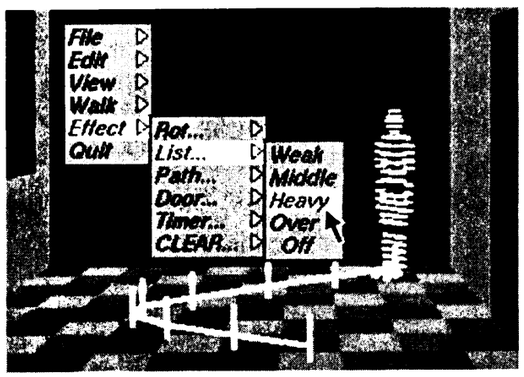


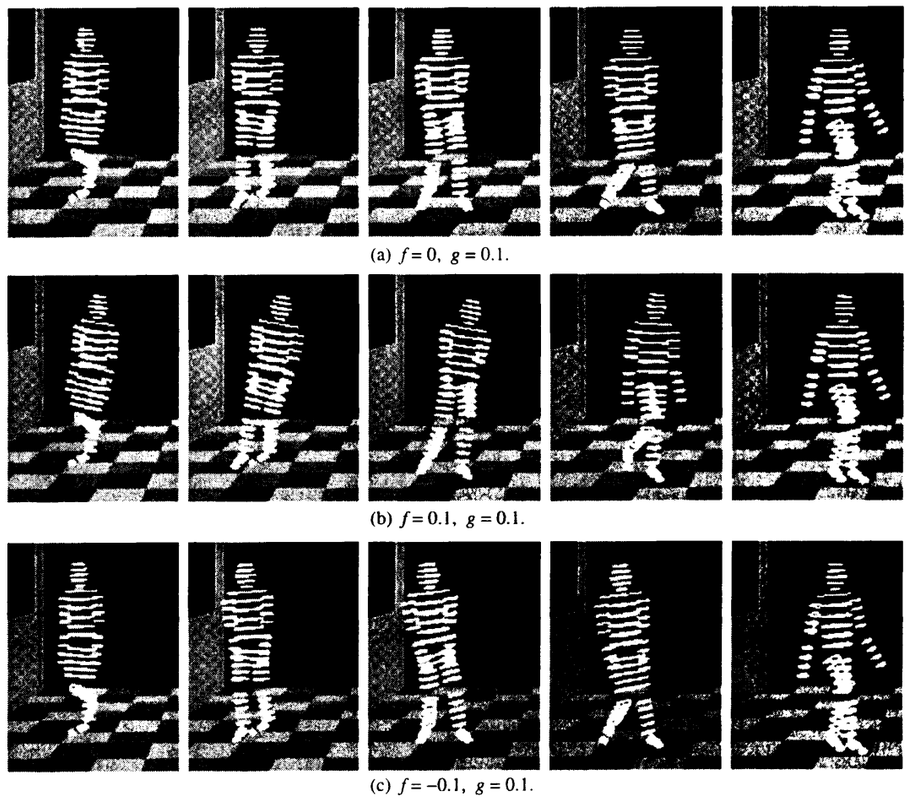
図 3.36: 誇張表現の因子設定例.

図 3.34 は人物像モデル設定の GUI 画面である．第 3.4 節と同様に，人物像の身長や体重，移動速度や標準歩幅を設定する．

図 3.35 は平面上の曲線歩行動作生成例である．図 3.35 (a) はパスの設定を，(b) はパスのステップへの分割を，(c) は分割後のパスに沿った歩行動作生成を示す．

図 3.36 は (3.38) 式や (3.42) 式における誇張表現因子 f あるいは g_{int}，g_{ext} の GUI による設定例を示し，図 3.37は，$g = g_{int} = g_{ext} = 0.1$ として f を変化させた曲線歩行の生成例を表している．

さらに図 3.38は，体軸を変化させた動作生成例を，人物像の上方に視点を設定して表示したもので，図 3.38 (a) は体軸を傾けないもの，図 3.38 (b) は体軸を回転の内側に傾けたもの，図 3.38 (c) は体軸を回転の外側に傾けたものである．これらの生成例から，誇張表現が良好に実現されていることが分かる．

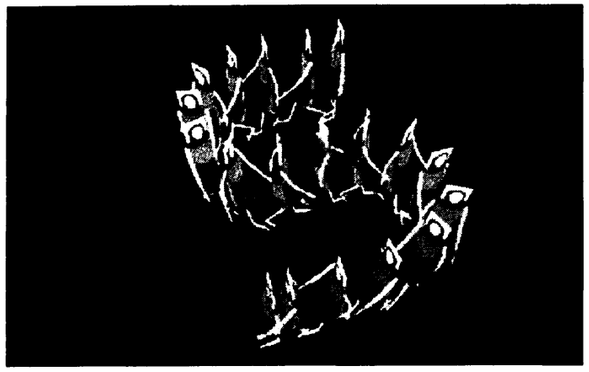


(a) $f = 0$, $g = 0.1$.

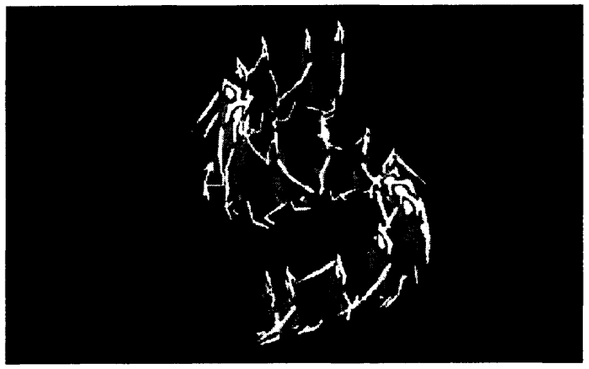
(b) $f = 0.1$, $g = 0.1$.

(c) $f = -0.1$, $g = 0.1$.

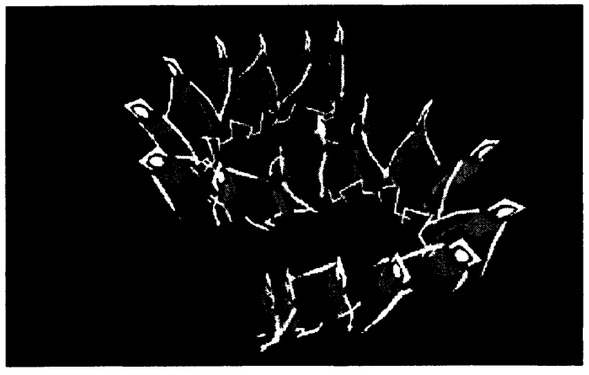
図 3.37: 体軸の傾きを変更して生成した例 (1).



(a)



(b)



(c)

図 3.38: 体軸の傾きを変更して生成した例 (2).

表 3.3: 1 歩に要する平均計算時間 (単位: 秒): (a) 直線歩行, (b) 平地曲線歩行.

	平地	階段昇	階段降	平均
計算時間	0.0364	0.0517	0.0533	0.0471

(a)

	運動学	動力学
計算時間	0.0448	0.0515

(b)

さらに，階段を下降して（人物像から見て）右へ方向転換する歩行動作生成例（$f = 0.1$, $g = 0.1$）を図 3.39に示す．これらの生成例により，地形変化及び曲線パスへの適応や誇張された歩行動作生成が実現されていることがわかる．

次に，3.3 (a) に直進歩行における平地並びに階段上昇，下降の動作生成に対する 1 歩あたりの平均計算時間（秒）を示し，3.3 (b) に平地での曲線歩行において，回転運動に運動学のみを用いた場合の計算時間と動力学を用いた場合の計算時間をそれぞれ示す．3.3 (a) の平地，階段上昇，階段下降の計算時間はそれぞれの動作を 10 歩生成させた時の計算時間を平均したものであり，「平均」はこれらの総合平均を示す．

これらの例において，歩調は全て 100 歩／分とし，フレームレートを 60 フレーム／秒（約 0.017 秒／フレーム）として一歩あたりのフレーム数を 36 とした．また，平面歩行における標準歩幅は 70.0 cm とした．

なお，この計算時間は，前述のハードウエア環境で行ったものであり，また，プログラムの最適化を行ったものではないため，絶対値ではなく相対値のほうが意味を持つ．これは，次節の検討においても同様である．

3.6.7 検討

生成例で示した通り，本手法は，一度歩行パスを決定すれば自動的にパスに沿った歩行動作を生成する．しかも，図 3.37や図 3.38で示したように，ユーザによる誇張表現の制御が可能である．

しかし，動力学計算が前後あわせて 3 歩の情報があれば生成できるのに対し，各ステップを連結し，誇張表現などを実現するためには，少なくとも同じモードの開始から終了までのパスあるいはステップが既知でなければならない．この問題を解決するために，例えば人物像モデルの物理的なバランスを考慮し，動力学計算を行う際に体軸の傾きなどを同時に計算することが考えられるが，歩行全体に対する調和やユーザによる制御，計算の効率化などを考慮し，本手法では運動学による関節角度等の修正によって制御する手法を採用している．

計算時間に関しては，表 3.3より，高速な計算を実現していることが確認できる．これは，運動

の分解の有効性を示している．また，表 3.3 (b) より，回転に動力学を用いた場合の計算時間は運動学のみを用いた場合の約 1.15 倍に留まり，効率良く動力学計算が行われていることがわかる．回転運動は実際には多くの要素が複雑に関係しており，より自然な動作を生成するにはさらに洗練された方程式を立てる必要があるが，本手法は回転動作を 1 変数の運動方程式により効果的に生成できており，計算効率を考慮すると十分と判断できる．これらの計算例では，運動学的計算では各フレーム間を分割せずに線形に補間しており，動力学計算は各フレーム間を 5 ～ 10 に時分割して行っている．

本手法では，第 3.4 節の直進歩行動作生成と同様に，各ステップの終了時（踵着地の瞬間）に次の一歩の計算を実行するため，本節で述べたハードウエア環境による計算においては，例えば描画速度を 60 フレーム／秒とする場合等は表 3.3 からわかる通り踵着地の瞬間に若干の遅延が生じることとなるが，実用上十分な計算速度である．

また，映像表現という観点からみた場合は，キャラクタの個性を表現するために，Perlin らの手法 [76] や文献 [96] に見られるようにユーザによる誇張表現等の制御が可能である方が望ましい．本手法では，関節角度の可動範囲を決定したり，図 3.36で示したように，3.6.5節で述べた境界条件や体軸の傾きなどを決定する表現因子等をユーザが対話的に設定して (3.38) 式や (3.42) 式に適用することにより誇張動作の表現が可能である．しかし，ユーザの意図する誇張表現と動作の自然さとを調和させるには，これらの運動学的な誇張表現が実際の歩行動作に及ぼす影響について考慮する必要があり，今後の検討課題のひとつである．

3.7 本章のまとめ

本章では，与えられた 3 次元仮想環境内に設定された任意の 3 次元移動経路に沿う歩行動作を自動的に生成する手法について述べた．

まず，高低差ある地形形状に設定された歩行パスに対応した直進歩行動作を，自動的に生成する動作生成手法について，その定式化と計算方法について述べた．本手法は，Bruderlin の KLAW システムを，凹凸ある地形にも適用可能なように拡張したものであり，実際に歩行動作の生成例を示すことにより，その有効性を確認した．

次に，人物像歩行動作に付随する，動力学モデルに基づく腕の運動の生成方法について述べた．腕，肩，肘は脚と異なり，地面に接したり，地形に対応するような拘束条件がない．そのため，運動に体する自由度が大きくなり，計算量の増加や不安定性が生じる恐れがある．このため，腕全体をひとつの物理振子で近似した運動モデルを適用し，肘角度を運動学的に補間する手法を適用した．本手法は比較的容易に荷重を含む腕の運動を生成することが可能であり，また，誇張表現などを目的とした肘角度の可動範囲の指定も可能である．

さらに，曲線歩行を含む，任意のパスに沿った人物像の歩行動作を自動的に生成する動作生成手法について詳細を述べ，その実現例と有効性を報告し，考察を行った．即ち，歩行を動力学部分と運動学による連結，誇張表現とから構成し，前者において歩行動作を直進歩行動作と回転動作に分

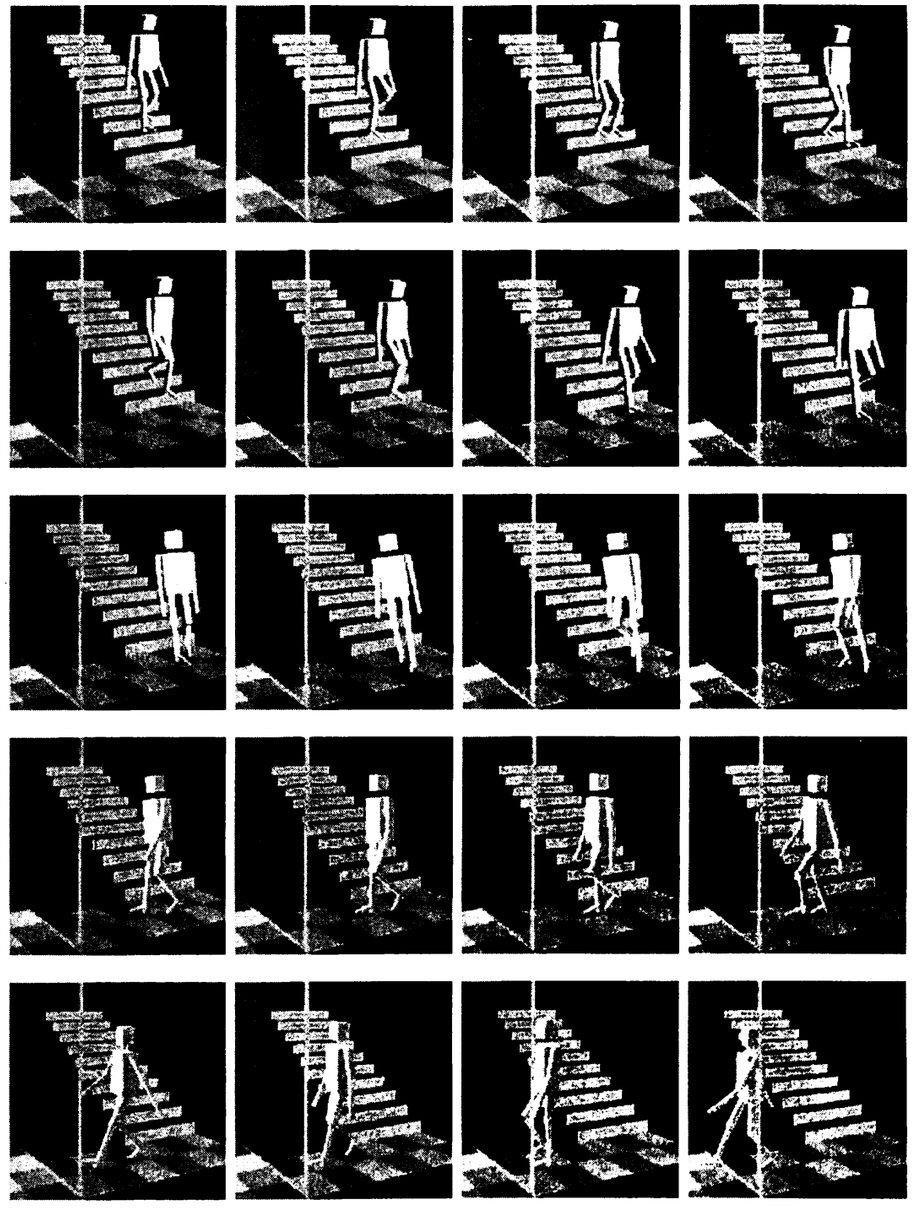


図 3.39: 階段を下降しながら方向転換を行うアニメーションの生成例．

解して生成し，後者において歩行パスやモードに適応するように各種境界条件等を計算してそれぞれを融合させる手法を用いた．

これらの計算は，ほぼ実時間で可能であり，また，動作を生成するための入力パラメータは，人物像の身長や体重などを除けば，基本的に移動速度のみで良いことを示した．

本章で述べた手法を用いることにより，歩行動作生成の負担が減少し，室内や建築物等の 3 次元環境モデルと歩く人物像とを融合することが可能となり，魅力的で活気的な映像や，リアリティあるシーンを生成することができるようになる．また，本手法は歩行動作を自動生成するのみならず，誇張表現等のユーザ制御も可能であるため，映像コンテンツの生成にとって極めて有効である．

第 4 章

歩行動作の効率的な生成および制御

4.1 はじめに

本章では，歩行動作の生成や制御，ならびに生成結果の表示を効率良く行う手法，即ち，個々の歩行者の動作生成や制御を効率よく行う手法と，多人数の歩行者の動作生成ならびに表示を効率よく行う手法について詳述する．

本章では，まず，前章で構築した歩行動作の運動方程式に対し，外力項に床反力を近似した関数を用いる手法を提案する．これは，人体の重心位置の変化に直接関係する床反力を解析関数の一次結合で表すことにより，数値計算上の安定と高速化をはかり，また，重心位置の直接的な制御を可能とすることを目的としている．

次に，多数の人数の生成を目的とする際に，各人物像を運動の更新周期の異なる複数のグループに分類して生成する手法を提案する．これは，多人数の歩行動作の生成を実現する上で，視点からの距離に応じて運動の更新周期を変化させることで，見た目の自然さを損なわずに全体の計算時間を減ずることを目的としている．本章では，高速な描画を必要とする応用例として，3 次元仮想空間における多人数の歩行者の生成および表示を対象としながら，アルゴリズムと実現例について述べる．

本章の研究成果により，個々の歩行者の歩行動作の計算を高速に，かつ，安定して行うことが可能となり，また，歩行状態の直感的な制御を行うことが可能となる．さらに，多人数の運動生成ならびに表示に要する時間を軽減することが可能となり，実時間性を要求するアニメーション・システムへの適用が可能となる．

4.2 床反力を用いた歩行動作の生成

本節では，人間が歩行する際に足が地面から受ける反力，即ち床反力に関する解析と，それを用いた歩行動作生成について述べる．本節でも，これまでの章と同様，パスに基づく歩行動作生成のアプローチに従うが，本節では，歩行動作を生成する外力として，新しく床反力を導入する [107]．

即ち，第3章における直進歩行動作生成（第3.4節），腕の運動の生成（第3.5節）回転運動等の生成（第3.6節）における動力学の運動方程式に現われる外力の項に，床反力を近似した解析関数を適用する．その理由として，

- 歩行動作を生み出す主な力は床反力であり，床反力に基づくアプローチは自然で合理的である，
- 床反力は身体の重心位置に直接関係しているので，a床反力と，生成結果 (となる重心位置) との関係が把握しやすい，
- 床反力を扱うことにより，重心位置の計算を容易に行うことができ，関節角度のトルク計算の際に生じ得る逆問題を回避できる，

という点が挙げられる．本節で述べる手法のアルゴリズムは以下の通りである：

1) 床反力を解析関数の一次結合として近似する，
2) 直進前進歩行，回転運動，腕の運動を，近似した床反力を用いて計算する，
3) 計算後の動きを補正する．

このアルゴリズムは高速で，かつ，自然に見える歩行動作を生成する．本手法を用いることにより，ユーザは3次元でモデル化されたシーンに対応する歩行動作を，3次元歩行パスを設定するだけで，リアルタイムで得ることができる．

4.2.1 床反力の分析

物理的には歩行動作は立脚に対する地面からの反力によって生成される．直進歩行に関しては，床反力の詳細な解析がされている [86][83]．図 4.1は，片方の足の踵着地から爪先が離れるまでの，直線歩行における典型的な床反力を示している．図 4.1において，“vertical” は鉛直上向きの床反力であり，“progressional” は歩行の進行方向の床反力である．“lateral” は左右方向の床反力であり，足の内側向きが正の方向である．

歩行の持続時間を床反力との関係を含んで改めて記すと，以下の通りである：図 4.2に示すように，片方の足の床反力の鉛直成分には2つのピークがある．ひとつは他の足の爪先離地（toe off）の瞬間であり，もうひとつは他の足の踵着地（heel strike）の瞬間である．持続時間には，既に第3章で述べた通り，

$$T^{stance} = T + T^{ds}, \tag{4.1}$$

$$T^{swing} = T - T^{ds}, \tag{4.2}$$

という関係がある．T は歩幅 w と移動速度 v から

$$T = w/v, \tag{4.3}$$

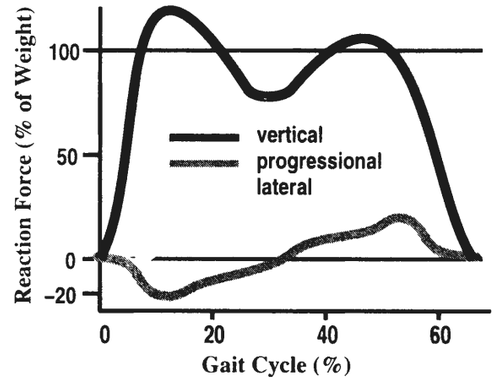


図 4.1: 1歩に対する典型的な床反力．

によって，あるいは，単位時間内の歩数 ν を用いて

$$T = \nu^{-1}, \tag{4.4}$$

から得ることもできる．本節でも，第3章と同様に，$T^{ds} = 0.25T$ という近似を用いている．歩行は，歩幅や速度などが変化しなければ，周期 T の周期的な運動とみなすことができる．

床反力を時間の関数で書き表すことができれば，人体の重心 $\boldsymbol{r}$ の運動方程式は以下のように近似できる：

$$M\ddot{\boldsymbol{r}}(t) = \boldsymbol{F}_L(t) + \boldsymbol{F}_R(t) - M\boldsymbol{g}, \tag{4.5}$$

ここで，M は人体の全質量である．$\boldsymbol{g}$ は重力加速度であり，$\boldsymbol{F}_L(t)$，$\boldsymbol{F}_R(t)$ はそれぞれ左足，右足が受ける床反力である．

本節では床反力を時間の関数として近似するために，直線歩行と曲線歩行，階段の上昇と下降における床反力を実際に測定した．ただし，用いた床反力測定装置は鉛直方向のみしか測定できないものである．被験者は20代の女性2名と30代の男性2名で，歩行開始や停止ではない途中の歩行において，5歩分の測定をひとつの歩行パターンにつき10度繰り返した．図 4.3はその測定の平均を表している．測定にあたり，曲線歩行に関しては，左足支持で左方向への方向転換と，左足支持で右方向への方向転換とを測定した．また，階段は幅が30cm，高さが16cmである．平均を取った理由としては，階段や平地などの歩行の際の床反力に対して，測定者に依存しない共通な特徴を抽出することができれば，「標準的な」歩行動作生成のための方程式を定式化できるためである．

この測定実験により，次のような特徴を得た：

1) 全ての床反力は2つのピークを同じタイミングで有する2つのピークを有する，
2) 直線歩行と曲線歩行では，それほど差異が見られない，
3) 平面歩行，階段上昇，階段下降において，2つのピークの高さに差異が見られた．

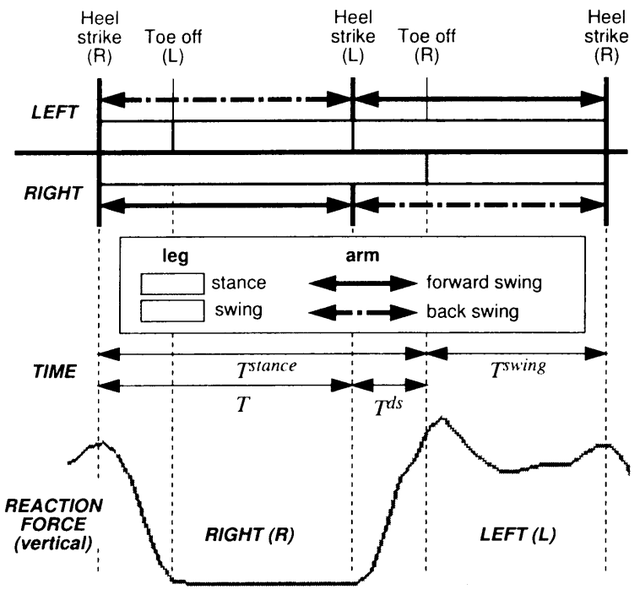


図 4.2: 持続時間と床反力.

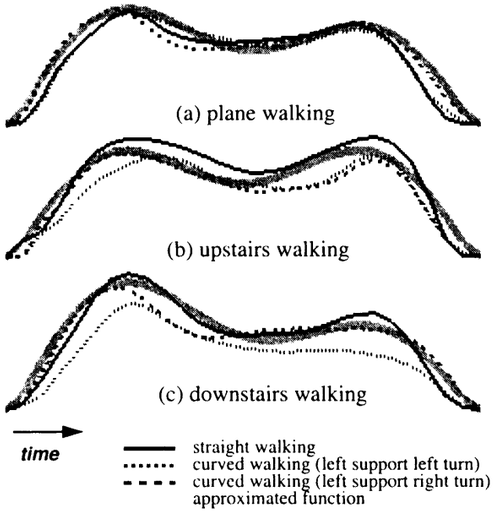


図 4.3: 鉛直方向の床反力測定結果と近似曲線.

以上の結果から，歩行の前進動作と回転動作は，鉛直成分に関しては互いに独立であると仮定した．これは，少なくとも測定した床反力の鉛直成分に関しては，曲線歩行も直線歩行も同じ傾向を見せたことにより，第3.6節の動力学の分解を裏付けている．また，これは歩行動作生成の方程式を簡単にし，計算コストを減少するのに有効である．

さて，これらの実験果から，左右の足に加えられる床反力は同一であるとみなし，床反力を以下のように近似した：

$$F_x(t) = s_x \sum_{k=1}^{3} a_{xk} \sin\left(\frac{2k\pi}{T^{stance}} t\right), \tag{4.6}$$

$$F_y(t) = s_y \sum_{k=1}^{3} a_{yk} \sin\left(\frac{k\pi}{T^{stance}} t\right), \tag{4.7}$$

$$F_z(t) = \pm s_z \sum_{k=1}^{3} a_{zk} \sin\left(\frac{k\pi}{T^{stance}} t\right), \tag{4.8}$$

ここで，$F_x(t)$ は床反力の進行方向成分，$F_y(t)$ は鉛直方向成分，$F_z(t)$ は左右方向への成分である．また，s_x，s_y，s_z はスケール因子であり，(4.8) 式において，$+$ は左足，$-$ は右足を表している．

前後方向の床反力 $F_x(t)$ に現われる係数については，以下のように近似した：

$$a_{xk} = -0.14286/2^{k-1} + \delta_{xk}, \ (k = 1, 2, 3) \tag{4.9}$$

ここで，δ_{xk} は床反力の個人差を表す因子で，インプリメント上はゼロで初期化してある．同様に鉛直方向にの床反力 $F_y(t)$ に現われる係数についても，以下のように近似した：

$$\begin{aligned} a_{y1} &= 1.3 + \delta_{y1}, \\ a_{y2} &= f_v + \delta_{y2}, \\ a_{y3} &= 0.4 + \delta_{y3}, \end{aligned} \tag{4.10}$$

ここで f_v は，平地歩行では $f_v = 0.1$，階段上昇では $f_v = 0$，階段下降では $f_v = 0.25$ である．δ_{yk} も，個人差を表す因子である．また，左右方向の床反力 $F_z(t)$ に現われる係数についても，δ_{zk} を個人差を表す因子として以下のように近似した：

$$\begin{aligned} a_{z1} &= 0.07 + \delta_{z1}, \\ a_{z2} &= -0.03 + \delta_{z2}, \\ a_{z3} &= -0.03 + \delta_{z3}. \end{aligned} \tag{4.11}$$

これらの係数の値は，測定したデータに対する最小二乗法によるフィッティングで求めたものである．図 4.4 のグラフは，(4.6) 式，(4.7) 式，(4.8) 式によって近似された床反力関数であり，測定結果の平均値を非常に似た傾向を示す．この近似関数を用いて，実際に人体の姿勢を計算することになる．

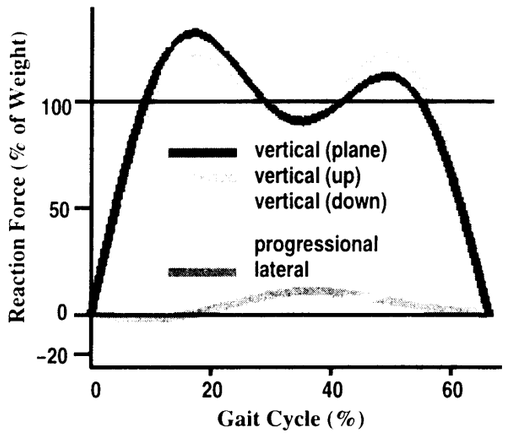


図 4.4: 床反力近似関数.

動力学計算においては，直線歩行動作は前後方向の床反力 (4.6) 式と鉛直方向の床反力 (4.7) 式とから生成される．ただし，前後方向と左右方向の床反力は測定できなかったため，実測値ではなく文献 (例えば [86]) に記載されている床反力の形状を基に近似を行った．このため，(4.6) 式および (4.8) 式によって表される床反力，そして (4.9) 式，(4.11) 式によって表される係数は，全ての地形パターンに共通して用いるものとした．

4.2.2 床反力を用いた歩行動作生成

本研究では，第 3 章において歩行動作を 3 つの動きに分解した：直進歩行動作，腕の運動，回転運動である．そして，これらお運動を互いに動力学的に独立であるとみなし，計算を行った．

動作生成の運動方程式は基本的にこれまで述べた手法と同様であるが，直進歩行動作生成については，外力に床反力近似関数を用いることにより，運動方程式自体もより簡略化することができる．腕の運動，回転運動については，それぞれ第 3.5 節，第 3.6 節と同様であるが，外力に前節の床反力近似関数を用いている．以下，それぞれの運動について説明する．

直進歩行動作の生成

直進歩行動作に関しては，まず，進行方向と，地面に対して鉛直な方向とから形成される鉛直面内での歩行動作が計算され，それから進行方向に対して左右である方向の運動が計算される．鉛直面内での運動の計算は以下のような手続きで行われる：

1) 身体の重心位置を動力学的に計算する，

2) 立脚の角度を内挿する，

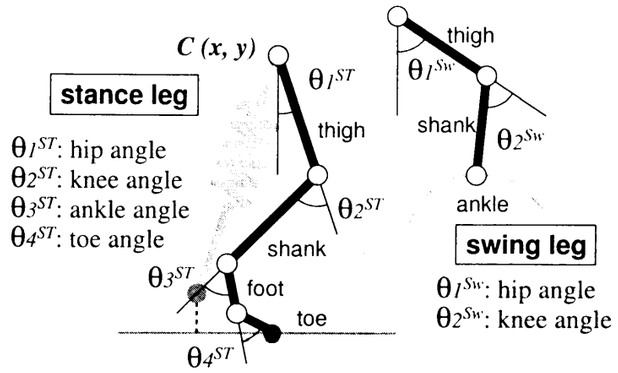


図 4.5: 脚部の動力学変数.

3) 遊脚の角度を動力学的に計算する，

4) 残りの角度を運動学的に計算する．

脚部のパラメータを図 4.5 に示す．ここで，$\boldsymbol{C}(x, y)$ は人体の重心位置を示す．

まず，(4.6) 式，(4.7) 式における $F_x(t)$，$F_y(t)$ を用いて，(4.9) 式，(4.10) 式の係数を適用して $\boldsymbol{C}$ の (x, y) 成分を計算する．計算において，(4.6) 式，(4.7) 式のスケール因子 s_x と s_y は $\boldsymbol{C}$ が初期位置から時間 T 後に終期位置に達するように反復して計算する．前節で述べたように，床反力関数と係数は固定した幅と高さを持つ階段を用いて測定した結果を用いたが，本節ではこの関数と係数を全てのパターンで利用した．T^{ds} の間は，床反力は両足の合計値であるため，ステップの開始から T^{ds} の間は，他の足の床反力を加える必要があり，これは直前の一歩のための計算値を用いている．

角度を計算するためには，おそらくはトルクを外力として用いたほうが物理法則にかなっている．しかし，本章で床反力を用いたのは，単に計算上の問題が生じにくいということだけでなく，床反力というものが歩行者の個性を反映しており，また，歩行動作生成を行うユーザが力を直観的に制御できるからである．つまり，トルクは直観的に把握しにくい概念であるが，力は把握しやすい概念であると言える．

さて，計算の初期値，終期値を計算するために，図 4.6に示すように，踵着地時の重心の位置がちょうど歩幅の半分である，という仮定を設ける [99]．図 4.6において，$\boldsymbol{C}_{iI}$，$\boldsymbol{C}_{iF}$ はそれぞれ i 番目のステップにおける重心 $\boldsymbol{C}$ の位置である．これにより，$\boldsymbol{C}$ と足首の位置の境界値が決定される．後に示すように，この足首位置の境界値を用いて，踵着地時の踵の位置を決定することができる．境界値において，速度や角速度は保存される．

立脚および遊脚の角度は，図 4.5 に示す通りである．$\boldsymbol{C}$ が決定されれば，大腿部の角度 θ_1^{ST}，膝角度 θ_2^{ST}，足首角度 θ_3^{ST}，そして爪先角度 θ_4^{ST} を内挿により決定できる．図 4.7 はこの内挿を示し

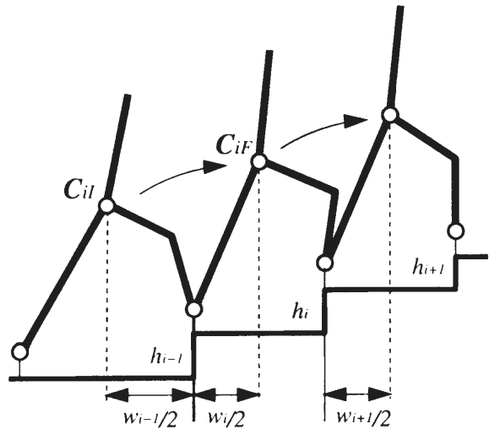


図 4.6: 直進歩行に対する境界条件.

ている．図 4.7 (a) は踵着地の瞬間である，ここでは膝の角度はゼロと仮定している．図 4.7 (b) は $\boldsymbol{C}$ が足首位置の真上に来た瞬間である．図 4.7 (a) から (b) まで，立脚と爪先は (図 4.5 も見よ) 固定された踵の周りを一定の角速度で回転するものと仮定し，θ_4^{ST} は固定されている．踵の位置は，図 4.7 (b) に示す足首の位置，即ち，ステップの境界 (図 4.6を見よ) における位置から計算することができる．階段歩行においては，爪先と踵が同時に着地するものと仮定し，足首と爪先の位置は (b) まで固定されているものと仮定する．

図 4.7 (b) から (c) までの間，立脚の足 (図 4.5を見よ) は固定された点 $\boldsymbol{M}$ の周りを滑らかに回転する，即ち，θ_4^{ST} はあらかじめ設定された一定の角速度で増加する．最後に，図 4.7 (c) から (d) の間，足と爪先が固定された爪先の周りを回転し，この間，爪先位置と θ_4^{ST} は固定されており，回転の角速度はやはりあらかじめ計算されている．

全ての期間において，$\boldsymbol{C}$，足首の位置，θ_4^{ST} は既知であり，大腿部，下腿部ともに長さが一定であるので，θ_1^{ST}，θ_2^{ST}，θ_3^{ST} を計算することができる．

図 4.5に示す遊脚の角関節角度 θ_1^{SW}，θ_2^{SW} は，第 3章と同様の方法で計算される [98]:

鉛直面での計算が終了したら，左右の動き即ち $\boldsymbol{C}$ の z 成分が計算される．これは (4.8) 式における力 $F_z(t)$ を係数 (4.11) 式を用いて計算し，変化が生じた関節角度を運動学的に計算する．図 4.8 は一歩での直線歩行での，重心の状態を示している．このグラフは，実際の測定値とよく合致している [86][83].

回転運動

回転運動は立脚の足首から鉛直上向きに伸びる軸まわりの回転によって生じる．既に第 3.6 節において回転運動に関しては述べており，ここでは繰り返さない．本節では外力の関数のみが異るので，それについてのみ言及する．

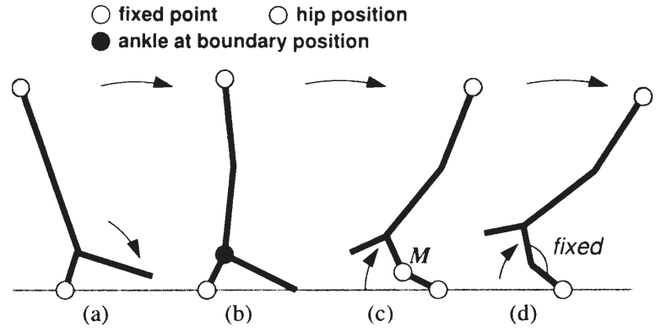


図 4.7: 立脚の補間（再掲）; (a) 踵着地, (b) 腰が足首の真上に来た瞬間, (c) 他の足の踵着地, (d) 爪先離地.

第 3.6 節で述べた回転に対する運動方程式に適用する外力として，上記モデルの基本として，本節では前後方向の床反力に基づくステップ関数を用いている (図 4.9参照)：

$$F_r(t) = \begin{cases} 0, & 0 \le t \le T^{ds},\ T/2 \le t \le T, \\ f_r, & T^{ds} \le t \le T/2. \end{cases} \tag{4.12}$$

その理由として，

1) 回転力が加わるのは，片脚支持開始にはじまり，前後方向の床反力がゼロとなるときに終えるものと仮定し，また，
2) これは高速で簡単な計算を実現するからである．

f_r の値は，f_r の値を変えながら，図 3.4における $\vec{d}$ を T の間に α だけ回転するような値になるように (4.12) 式を反復計算する．この値は左への方向転換では正であり，右方向では負である．

腕の運動

腕の運動についても，外力についてのみ言及する．本節では，腕の運動に対する運動方程式のトルク項も，腕の運動の方向が前後方向に平行であることから，前後方向の床反力に基づくステップ関数で近似する (図 4.10参照).

$$F_a(t) = \begin{cases} f_a^1, & 0 \le t \le T^{ds}, \\ f_a^2, & T^{ds} \le t \le T^0, \end{cases} \tag{4.13}$$

ここで T_0 は前後方向の床反力の値が 0 になる時刻である．f_a^1 は加速のトルクであり，f_a^2 は減速のトルクであり，これらの値は，第 3.5 節で述べた境界条件を満足するように，これらの値を変更しつつ (3.26) 式を反復して計算することによって求める：

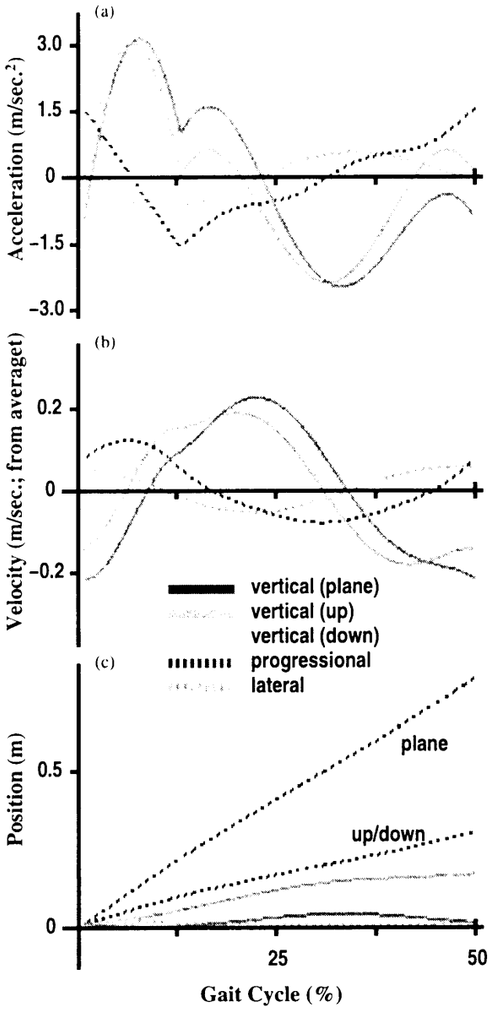


図 4.8: 人物像重心の物理状態; (a) 加速度, (b) 速度の平均移動速度に対する割合, (c) 位置.

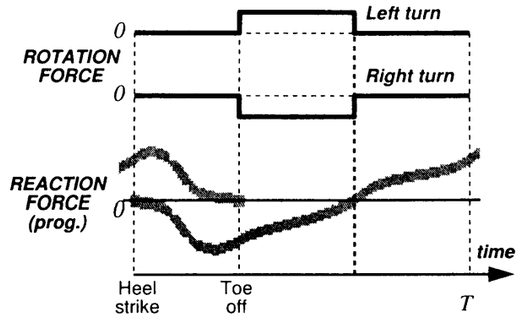


図 4.9: 回転運動における一般化力関数の近似.

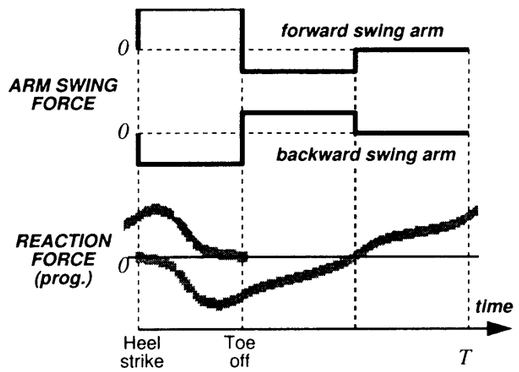


図 4.10: 腕の運動における一般化力関数の近似.

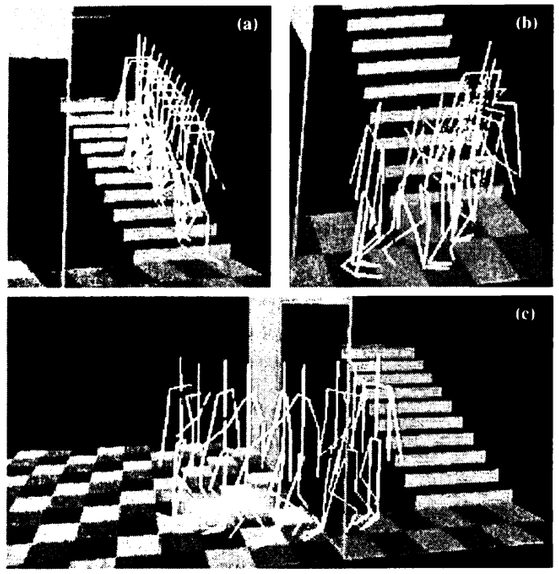


図 4.11: 生成結果例; (a) 階段の下降, (b) 階段の上昇, (c) 平地での曲線歩行.

4.2.3 生成結果例と検討

図 4.11，図 4.12は生成した歩行シーケンスを示す．これらの図から，本手法によって，3 次元でモデル化されたシーンに適合する自然な歩行動作を生成していることがわかる．

本節の結果は OpenGL^{TM} で実装され，シリコングラフィックス社製の OCTANE^{TM} (R10000 / 195 MHz) で実行された．パラメータの値は $\nu = 100/min.$，即ち 1 分あたり 100 歩であり，1 秒あたり 60 フレームとした場合，実時間での計算を得ることができた．さらに，歩幅が小さい場合でも，数値的な不安定さを生ずることなく計算を実行することができた．

本節において，

1) 直進方向と左右方向への床反力を，全ての歩行パターンに対して，平地歩行に基づくデータ [86] を用いて近似した，それは，鉛直方向の床反力しか測定できなかったためである．

2) また，回転運動や腕の運動に対する力やトルクを，前後方向の床反力に基づいて近似した，それは，回転や腕の運動を生ぜしめる力やトルクを実際には測定できなかったからである．

3) 広範な地形に適合する床反力の係数を計算するのに，反復計算を用いた．これは，階段上昇や下降のための鉛直床反力近似は，一定の幅と高さを持った階段での測定結果に基づいているため，直接適用するのではなく反復によって境界条件に合致させる必要があるからである．

さて，結果のアニメーションの品質 (図 4.11や図 4.12) により，我々の床反力の近似と，3 つの独立な部分運動に分割した歩行動作の生成モデルが適切であることが確かめられる．

アニメーション生成において，さらに洗練された運動生成の観点からすると，キャラクタの個性を扱うことが非常に重要となって来る．本節では床反力の近似において平均的な値を用い，δ_{ik} $(i = x, y, z;\ k = 1, 2, 3)$ の値を 0 に設定した．つまり，没個性的な運動となっている．従って，今後は個性的な歩行動作の生成と δ_{ik} との関係を解明することが重要である．

また，本節では，鉛直方向の床反力のみを扱っていたので，前後方向や左右方向も，様々な地形パターンに対して測定し，解析する必要がある．そして，回転運動や腕の運動に要する力やトルクと，それらの関係を解明して行かなければならない．さらに，運動の質を向上するために，腕の運動や回転運動の影響を考慮したバランス項を (4.5) に加える必要があるだろう．

それにもかかわらず，本手法は単に歩行パスの制御点を入力するだけで 3 次元歩行パスに適合する歩行動作を生成するという本研究の特徴を保持し，さらに，計算上の不安定さを解消して高速な計算を行い，アニメータの仕事量を削減するのに有効である．

4.3 更新周期制御による多人数の歩行動作の生成

本論文でこれまで述べてきたことは，ひとりひとりの歩行者の動作の生成についてであった．本節では，多数の歩行者を効率良くに生成する手法に付いて述べ，それを 3 次元仮想空間としての都市空間などにどのように適用するか，について議論する．

都市空間を電子的に構築し，様々な研究対象あるいは手段とするものとして，ディジタルシティがある．ディジタルシティは，研究やビジネスの分野のみならず，インターネットを介した社会的な活動のためにも新しいコミュニケーションの場を提供している．今日，ネットワーキング，社会活動，エージェントやコンピュータ・グラフィックスなど，様々な技術がディジタルシティのために研究・開発されている [47] [48]．このような仮想的な都市空間においては，人物像は重要な役割を果たしており，動き回る人物像を生成して表示することは非常に重要な問題である．

図 4.13は，このようなディジタルシティのひとつである「ディジタルシティ京都」プロジェクトにおいて作成された，京都市の祇園の 3 次元 CG モデルである [47]．本節の目的は，図 4.13 (b) のように多数の歩行者を生成し表示することである．しかし，多数の動き回るオブジェクトの表示は計算量の増大を招く．なぜならば，多数の運動するオブジェクトをアニメーションにより表示することは，各表示フレームごとに動く物体の状態の変化が計算され，表示されるからである．

本節では，この問題を解決するために，多数の歩行者をなるべく高速に生成・表示する手法に主眼をおいて述べる．これまで述べてきた歩行動作生成手法を応用するために，まず，一人の歩行者の生成と表示について述べ，その後，“群衆としての歩行者” の生成と表示について述べる．本節で述べる手法では，人物像を距離に応じてグループ分けし，視点から遠い距離にある人物像は，視点から近い距離にある人物像に比べ，運動状態の更新をより少なく実行する．本節ではそのアルゴリズムに付いて述べ，描画に対する実験結果を示す．

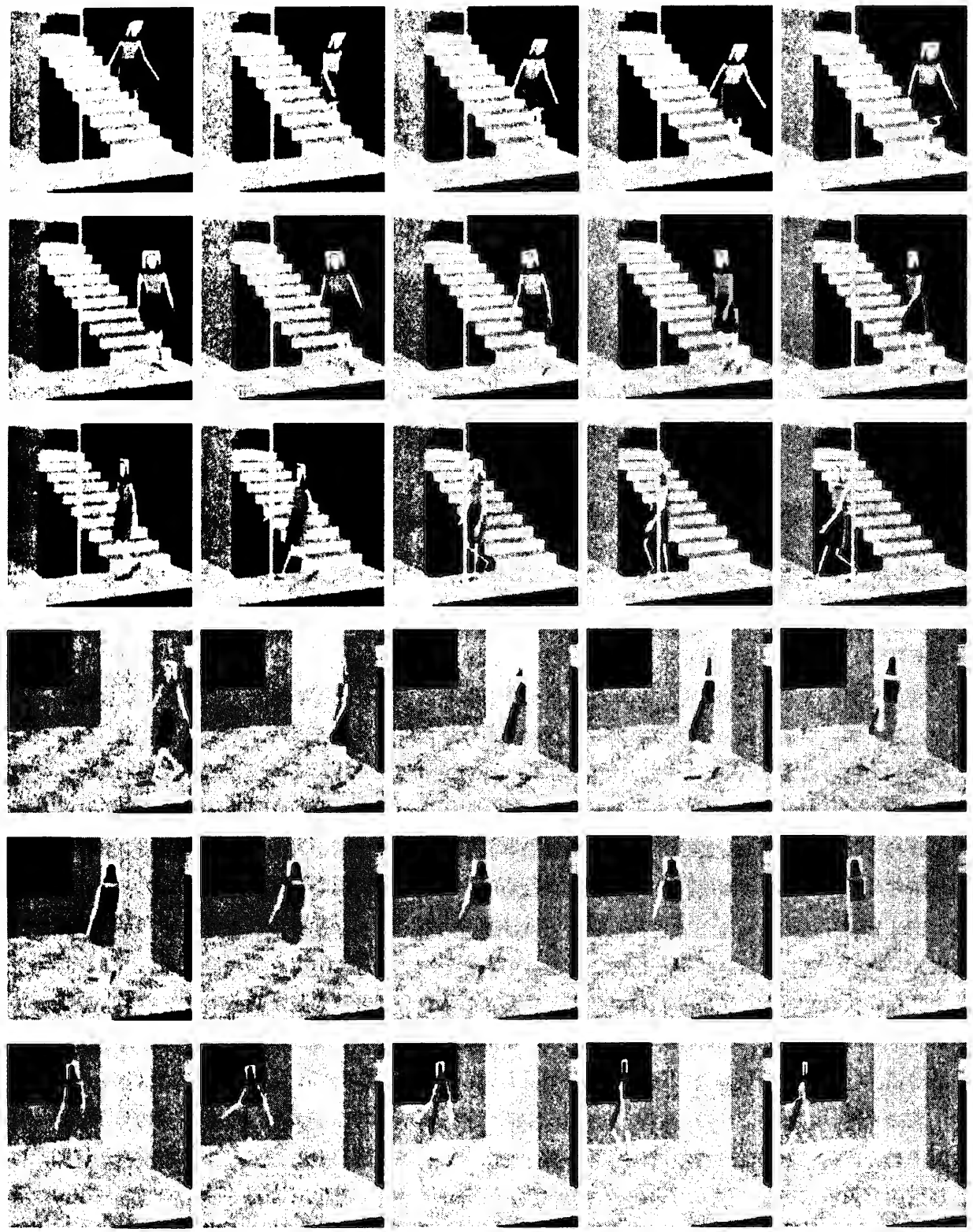


図 4.12: 生成結果例; 上段から下段へ, 左から右へ進む.

(a)

(b)

図 4.13: 京都市祇園の CG 画像; (a) “ディジタルシティ京都” からの画像, (b) 歩行者を加えたもの.

4.3.1 一歩行者の動作生成

多数の歩行者は，一人の歩行者の動作生成が基本となっている．後述するシステムの構築（第 5 章）の図 5.3 に示すように，一歩行者の動作生成の手順は以下のようなものである：

1) 3 次元環境データを読み込む．
2) 3 次元人物像モデルを読み込む．これは，身長，体重，デフォルトの歩幅などの情報を含む．
3) 歩行パスを設定する．
4) 歩行のための境界値を計算する：歩幅，高低差，回転角度，境界での身体の角度．
5) 各ステップに対し，以下を実行する：
   5.1) 制御パラメータ (移動速度) を入力する．
   5.2) ステップの持続時間，一歩あたりのフレーム数，一分あたり歩数を計算する．
   5.3) 一歩の動作を生成する．
   5.4) フレーム毎にシーンをアップデートし，描画する．
6) 結果の動作データ，あるいは動画を出力する．

これまで本論文で述べて来た通り，本手法は，リアルな歩行動作を非常に簡易な操作で生成することが可能である．第 3.6節で述べた手法を用いると，地形変化があるような場合，本手法では，全ての境界条件は前もって計算されていなければならないため，歩行の方向や状態（歩行開始，歩行中，停止など）を修正することができない．

しかし，仮想都市空間などに歩行者を生成するような場合，歩行状態や歩行の方向，速度などをユーザからの要求に応じて対話的に変更させることが必要となる．そこで本手法では，連続したステップにわたる誇張表現や制御などを行わないことにし，各ステップの独立性をより強化することで，第 3.6章で述べた手法を拡張した．

即ち，図 4.14 に示すように，各歩行者へのアニメーションは以下の手順に従う：

1) 歩行者に対する位置，方向，速度やモード，ユーザの視点などの情報を読む．
2) 状態が変更されていれば，前のステップの運動データを用いて 4) に進む．そうでなければ，次のステップの境界条件と移動パラメータを計算し，3) に進む．
3) 一歩の動きを計算し，運動データをメモリに記憶する．
4) 一歩の運動を表示する．
5) 1) に戻る．

この場合，方向転換や状態の変化をステップごとに制御することが可能となるが，そのような場合にはあらかじめ境界条件を計算することができないため，高低差があるような地形には適応が困難であり，今後対応策を考慮しなければならない．本章では，歩行領域を平地に限定することにより，方向の転換や状態の変化を対話的に行えることを可能としている．図 4.15 は，歩行者の視点から見たシーンである．

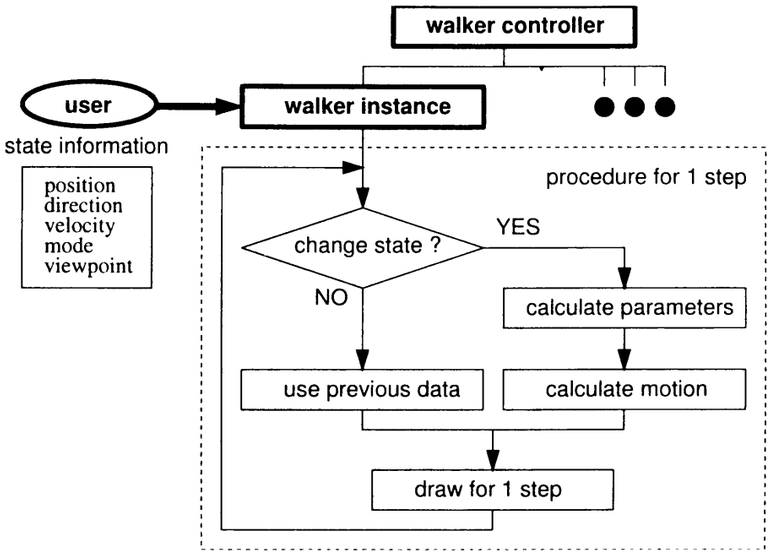


図 4.14: WALKER モジュール．

4.3.2 群衆としての歩行者

前述の通り，仮想都市空間に多くの歩行者が歩いていたら，都市空間のリアリティは飛躍的に増大する．しかしながら，多数の歩行者を生成し，表示するためのコストは非常に大きい．

コンピュータ・グラフィックスによる 3 次元シーンを描画するコストを最小化するための多くの手法が提案されている．もっともよく採用されている手法は，単純な幾何形状を用いて「表示」時間を減ずる手法である．よく知られている問題は，隠面消去（hidden surface removal）（例えば文献 [58] など）で，高品質なアニメーションの生成・表示には重要な手法である．現在もっとも多く採用されている手法は，LOD (Level Of Detail) と呼ばれている手法である．このアプローチは見た目の品質をなるべく保持したまま，ポリゴンや頂点の数を減らすものである．（例えば文献 [27][28][37][44][57] など）．

そのような，フレーム毎の描画時間の削減のみならず，動きを生成するための計算時間も何らかの指標を基に減らす試みがなされている．例えば Funkhouser らは，あらかじめ複数の運動状態を定義し，それらを視点からの距離，オブジェクトの速度，表示サイズなどを含んだ，経験に基づく基準で切替えている [36]．Carlson らはアニメーションに LOD 的なアプローチを適用した；これは，動作生成の手法を何らかの基準に基づいて切替えており，視点に近いものは動力学的なシミュ

図 4.15: 歩行者の視点から見たシーン.

レーションで生成し，ある程度視点から距離があるものは運動学的なモデルで生成し，視点から遠いものは質点モデルで運動を生成している [26].

これらの手法の全ては，シーン全体を全てのフレームで更新するものと仮定している．本節で述べるアプローチは，運動の状態や幾何的な情報を変更するのではなく，フレームレートや更新する周期自体を変更するというもので，新しい概念である．この新しいアプローチは上述の手法と相反するものではなく，組み合わせることでより大きな効果を上げることが可能である.

4.3.3 運動の削減

一般的に，図 4.16 に示すように，表示ウィンドウにおいて，視点から遠いオブジェクトはこれらに比較して近い位置にあるオブジェクトよりも，同じ移動量であっても表示される移動量が少ない．提案手法は，視点に近いオブジェクトは毎フレームで運動の更新を行い，一方，視点から遠いオブジェクトは周期的に運動を更新するものである．これは画像の更新に要するコストを軽減する.

この手法を実装するために，まず，歩行者は視点からの距離に応じてまずいくつかのグループに分割される．図 4.17 に示すように，各グループは固有の更新周期 P，領域および人数を有する.

全てのグラフィックス・オブジェクトは，その状態（位置，姿勢，形状，属性など）に変更が無くても，毎フレームごとに再描画されなければならない．そこで，本手法では，変更がないグラフィックス・オブジェクトを全てグラフィックス・ハードウエアのメモリに保存し，更新がない場合は，このメモリに記憶されたオブジェクトを呼び出して描画するアルゴリズムを用いる．図 4.18 はこの手順を示している：

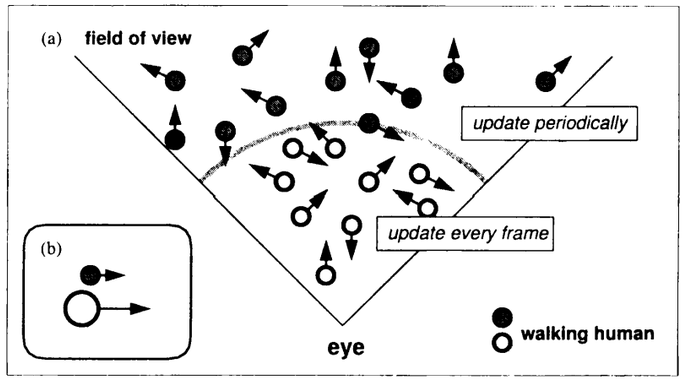


図 4.16: 本手法の概念; (a) 視界, (b) 表示ウィンドウ.

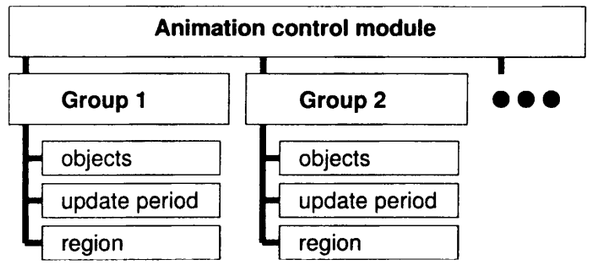


図 4.17: グループの概念.

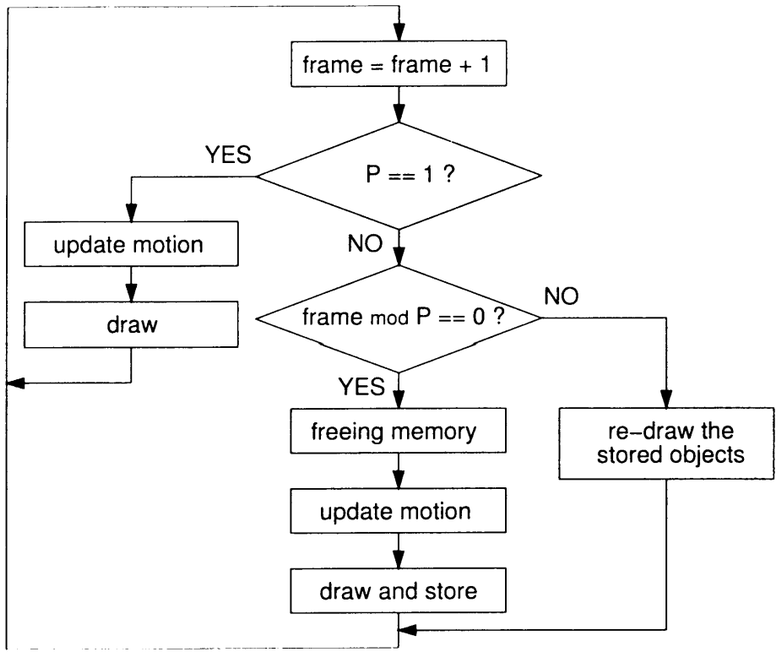


図 4.18: 提案手法のフローチャート.

1) フレームをインクリメントする.

2) $P = 1$ であれば，グラフィックス・オブジェクトを描画し，1) に戻る．そうでなければ，3) に進む.

3) (f mod P) $= 0$ であれば以下を実行し，そうでなければ 4) に進む.

3.1) メモリに記憶されたグラフィックス・オブジェクトを開放する.

3.2) 前回の更新から P フレーム分だけ更新する.

3.3) 更新されたシーンを描画し，メモリに保存する.

4) メモリに記憶されたオブジェクトを描画し，2) に戻る.

この手法は，メモリに記憶するグラフィックス・オブジェクトが多い程，即ち，同じ状態の人物像が多数存在すればより効果が現れる.

4.3.4 生成例と検討

前節のアルゴリズムの検証のため，背景画像に対して平面である歩行領域を設定し [103]，以下の条件を課して実験を行った：

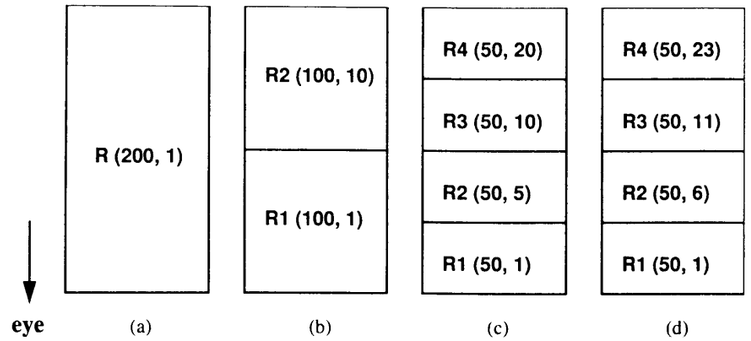


図 4.19: 実験したグループ例.

- **各グループ内の人数は保存される.**
  あるグループの歩行者が可視領域を離れた場合，そのグループに新しい歩行者が追加される.
- **全ての歩行者のポリゴン数，頂点数は同一とする.**
  ポリゴン・リダクションや隠面消去は適用しない.
- **動作生成の手法は，全ての歩行者で同一とする.**
  これまで述べた歩行動作生成手法を適用する.

本実験では，全可視領域に 200 人の歩行者を均質に分散して生成し，以下の条件を考慮しながら 100 フレームごとに計算時間を計測した.

1) **更新コスト.**
更新に要するコストを，レベル 1 からレベル 1000 まで変更した．このコストにおいては歩行者間の距離の計算などを実行し，数値は相対値である．即ち，比較のために，レベル 1000 においては，レベル 1 の計算を 1000 回繰り返している.

2) **更新周期.**
更新周期を 1(メモリに記憶しない状態) から 20(20 フレームごとの更新) まで変更した.

3) **グループ数.**
図 4.19に示すように，4 種類のグループ化を行った．この図において，グループは $R_i(N, P)$ で示されており，R は領域，N は歩行者数，P は更新周期である．人物像はいずれもこれらのグループのどれかに属する.

図 4.20 は上記の実験結果を示している．全ての計算および表示は，シリコングラフィックス社製の OCTANE^{TM} (R10000/195MHz) を用いて行った．図 4.20 (a) によると，ひとつのグループに 200 人を表示した場合，更新周期が増えると，負荷レベル (更新コスト) が高い時に特に，計算および描画速度は減少する．図 4.20 (b) によると，複数のグループ化は，負荷レベル (更新コスト) が高い時に特に，計算および描画の時間を有効に減らしている.

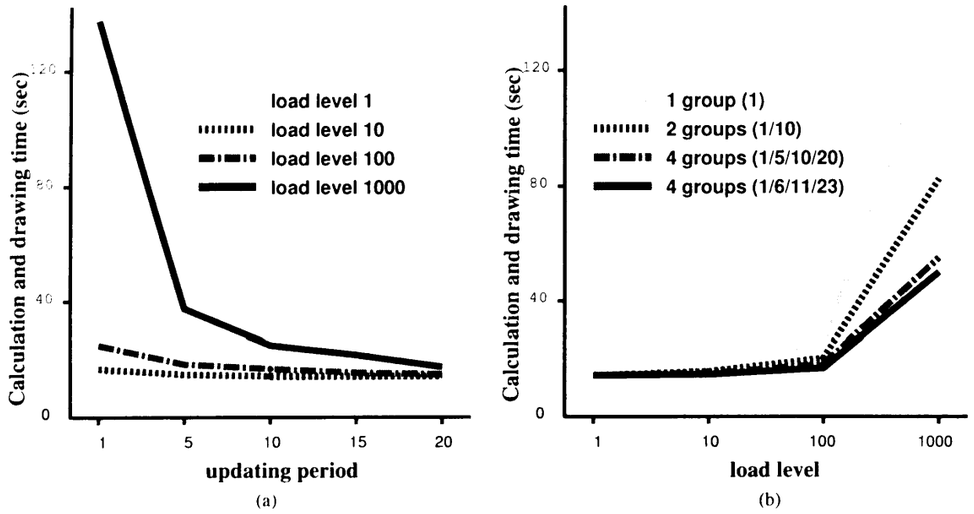


図 4.20: 200 人の歩後者に体する計算時間 (100 フレーム); (a) 1 グループ, (b) 各種のグループパターン.

1 グループのとき，もちろん，更新周期 P が大きいと動きの不連続性が生じる．例えば，更新周期が 20 になっていると，20 フレームの間は何も動かなく，20 フレーム後に，突然大きな移動量で移動する．グループの数を増加させて更新のタイミングをずらせば，この動きの不連続性を減ずることができる．

図 4.21 は，本手法を用いて表示した結果を示している．グループ化のパターンは，図 4.19(d) を用いている．この図から，オブジェクトの数は 30 倍に増加しても，計算および描画の時間は 20 倍しか増加していないことがわかり，本手法の有用性を示している．

ディジタルシティのようなインタラクティブなアプリケーションにおいては，例えばパーソナル・コンピュータを利用している場合，さらに非常に多くの他のタスクが生じるため，本手法は，これらのタスクがプロセッサの処理能力の多くを占めるような場合に，計算や描画に要するパワーを減ずるのに有効である．本手法を他の手法，例えば幾何データ減少法などと組み合わせることによって，さらに計算および描画時間を減ずることが期待できる．

本手法の応用例として，以下のようなディジタルシティへの適用を行うことが可能である：

- **ディジタルシティの開発や保守**:

これらに携わる人々は，ダイナミックなディジタルシティを，より少ないコストで設計できる．動く歩行者を制御するパラメータの量は少ないので，本手法はネットワーク・アプリケーションにも適している．さらに，本手法の運動削減手法は，他の動的なオブジェクト，車やバスや動物を含むシーンの表示にも適用できる．

図 4.21: アニメーションシーン (負荷レベル =1000), t は 100 フレームごとの計算時間; (a) 歩行者なし, (b) 10 人, (c)100 人, (d) 300 人.

図 4.22: 人気のある場所の表示例.

• **情報提供**:

ディジタルシティの開発者は，歩く人達を，何らかの情報を提示するものとして利用可能である，例えば宣伝，アクセスランキングや混雑状況，などである．図 4.22 は，人気のある場所を示している例である．

• **ユーザ**:

ディジタルシティのユーザは，歩行者としてディジタルシティ中を歩きまわることができ，また，人間型のツアーガイドが彼らをディジタルシティへの旅行へ案内することもできる．

4.4 本章のまとめ

本章では，個人及び複数の歩行者の歩行動作生成を効率的に生成し，描画する手法に付いて述べた．

まず，床反力を近似した関数を用いて，人物像の歩行動作生成を自動的に生成する手法について述べた．この法の特徴は以下の通りである：

1) 歩行動作生成の運動方程式に現われる外力の項に床反力近似関数を適用した，

2) 床反力似関数として，実測値と文献からの値を解析関数でフーリエ展開して近似した，

3) 逆問題を回避し，安定した数値計算を実現した．

本手法は，平地歩行，階段上昇や下降，曲線歩行を含む複雑な歩行動作に対し，少ないパラメータによる非常に簡易化された生成方法を提供する．床反力を用いたアプローチは安定で高速な計算を提供し，直接的に床反力関数の係数を処理することによって，広範囲な歩行動作の生成の可能性を秘めている．

次に本稿では，複数の歩行者を効率良く生成し，表示する手法について，特にディジタルシティへの応用を考慮しながら述べた．まず，一歩行者の歩行動作の制御に関して述べ，次に多数の歩後者を描画する際の計算コストを減ずる新しいアプローチを示した．このアプローチの主眼は視点からの距離に基づく周期的な更新である．そして，多数の歩行者を含むアニメーションに対する実験によって，その手法の有効性を示した．

本章の成果により，歩行動作の生成ならびに制御を高速化・安定化し，アニメータによる直接的な歩行上体の制御を可能とし，また，実時間性を要求されるディジタルシティのようなシステムへの応用が可能となった．

第 5 章

アニメーション・システムの実装および応用例

5.1 はじめに

本章では，第 3 章，第 4 章で述べた手法を実現し，様々な場面で応用するためのアニメーション・システムの構築について説明し，それらの適用例を示す．

まず，仮想環境に対し，与えられた歩行パスに沿う歩行動作を生成するためのアニメーション・システム *WorldWideWalk* の概要について述べる．歩行動作生成部分の処理の流れを解説した後，生成例で用いる人物像モデルについても説明し，実現例を示す．

次に，室内における歩行動作の生成において，歩行パスの設定に対する入力コストを軽減するための，反射モデルを用いた歩行パスの自動設定手法について述べる．さらに，奥行情報を持たない 2 次元背景画像と 3 次元歩行動作との合成について述べ，ネットワークを介したアニメーションの協調制作についても言及する．

その後，2 次元の人物画像を用いて，3 次元人物像モデルを作成し，それまでに述べた歩行動作生成手法を適用した例について述べる．

本章では，本研究で提案した歩行動作生成手法を，さまざまな映像コンテンツの制作に適用した実現例を示すことにより，映像制作現場における本研究成果の有効性を示す．

5.2 アニメーションシステム WorldWideWalk

5.2.1 システムの構成

本システムは，3 次元仮想環境に対して歩行パスを設定し，その歩行パス上を人物像モデルを歩行せしめる手法を提供することが主眼である．本研究成果を実施したアニメーション生成システムを，本章では *WorldWideWalk*（*WWWalk* と略記）と称する． *WorldWideWalk* は，

(1) 様々な環境コンテンツの適用，
(2) 簡易な入力・制御インタフェース，

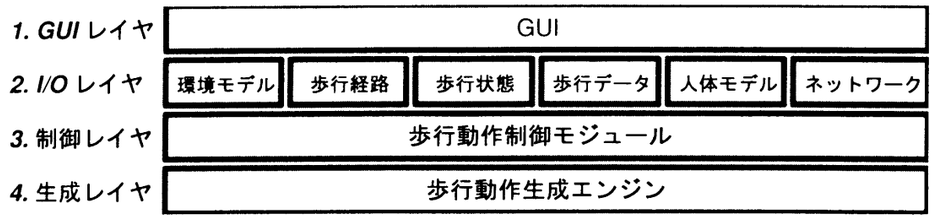


図 5.1: システムの階層構造.

を同時に実現することを目的としている．このような処理を適切に行い，また，汎用性を高めるためには，

- 環境コンテンツの種類や入力の手段に依存しないこと，
- 環境コンテンツに 3 次元歩行パスが設定可能であること，
- 設定された 3 次元歩行パスを一歩毎に分割し，適切な幾何的パラメータを計算すること，
- 一歩毎の幾何的パラメータに合致するように歩行動作を生成すること，

が必要となる（幾何パラメータについては後述する）．前述の条件を満たすために，*WorldWideWalk* は図 5.1に示すように，**GUI レイヤ**，**I/O レイヤ**，**制御レイヤ**，**生成レイヤ**の 4 層の階層構造をなしている．各レイヤは独立したモジュールとして構成されており，歩行動作の制御および生成は適用する環境コンテンツに依存しないようになっている．以下，それぞれのレイヤについて説明する．

GUI レイヤ

ユーザとの対話的な処理を行い，生成した結果を表示する部分である．視点を変更することにより歩行者から見たウオークスルーシーンを表示することも可能である．本システムではプラットフォーム間の互換性を保つよう，OpenGL^{TM} [115] および GLUT [50] を用いて実装されている．

I/O レイヤ

ユーザから，あるいはネットワークを介しての各種データや設定の入出力を行うモジュール群から構成される．それぞれのモジュールは以下のとおりである．

(1) 環境モデル

環境コンテンツの読込みを行う．本システムでは，OpenGL^{TM} で表示可能な 3 次元 CG，2 次元画像，奥行情報付き 2 次元画像が対象となっている．奥行情報を持たない 2 次元画像の場合は，第 5.4 節で述べる手法によって歩行領域を 3 次元平面として設定し，画像撮影時（あるいは描画時）のカメラ情報を復元することによって仮想的な奥行情報を付与することで 3 次元歩行パスの設定を可能としている．

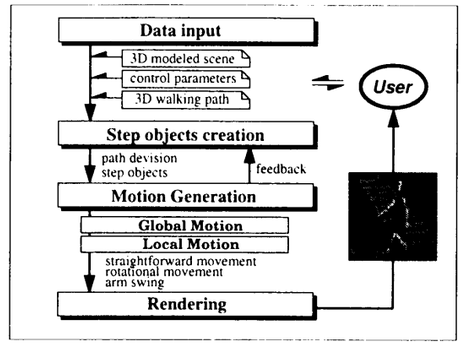


図 5.2: 歩行動作生成システムのブロック構成図.

(2) 歩行経路

環境コンテンツに対する歩行パスの設定処理を行う．第 3 章で述べたように，ユーザによって設定された 3 次元歩行パスは，人物像の歩幅や環境コンテンツを構成するポリゴンとの交点などから，一歩ごとに分割される．

(3) 歩行状態

歩行の速度や基本となる歩幅などの設定処理を行う．

(4) 歩行データ

生成された歩行データをファイルへ出力したり，あるいは記録されていた歩行データを読込む．ここで言う歩行データとは，人体モデルの重心位置および関節角度の時系列データである [101].

(5) 人体モデル

CG キャラクタなどの人物像モデルを読込み，構造化する部分である．本システムでは人物像の骨格部分を剛体棒で近似し，関節で接続された多関節構造体として近似しており，骨格部分に対して歩行動作の生成処理を適用する．人体モデルについては，5.2.3節で説明する．

(6) ネットワーク

ネットワークを介して複数人の歩行動作を生成する．第 5.5 節で述べるように，本システムでは，あるクライアントで生成された歩行データを他のクライアントにフレーム毎に転送して表示させたり，あるいは歩行パスに関する情報のみを特定のクライアントに転送し，歩行動作の生成と表示を同時に実行させたりすることが可能である．

制御レイヤ

歩行動作生成に必要な各種パラメータを計算する．このレイヤにおいて，ユーザによって設定された歩行パスを用いて，第 3 章で示した歩行時の幾何的パラメータや持続時間を計算する．また，本レイヤは環境コンテンツにおける人物像の位置や歩行状態も管理している．

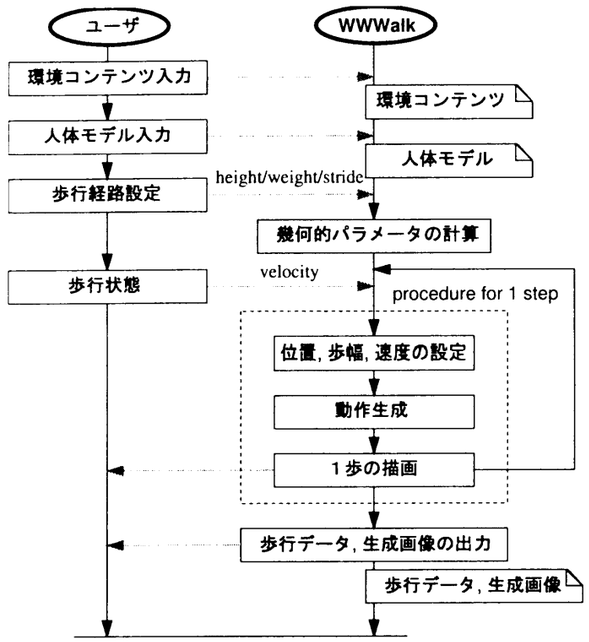


図 5.3: 歩行動作生成の処理の流れ.

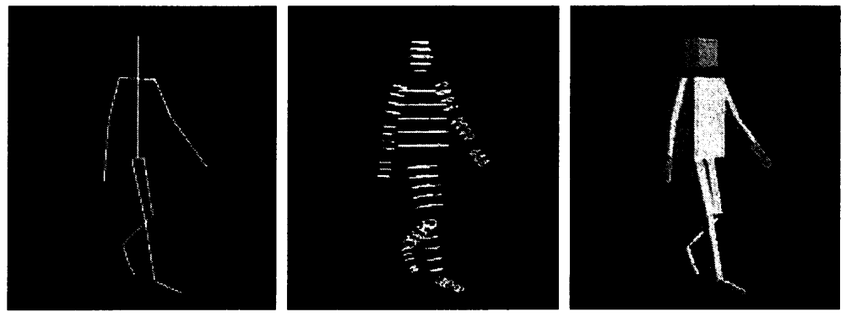
生成レイヤ

制御レイヤからのパラメータに基づき，多関節構造体で近似された人物像モデルに対し，歩行動作を生成する，本研究の最も重要な部分である.

5.2.2 WorldWideWalk における処理の流れ

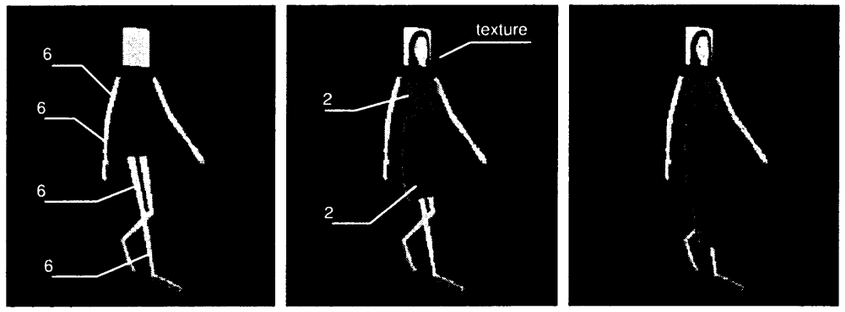
本システムにおいて，新しく歩行動作を生成する際には，ユーザからの入力に基づき以下に示すような処理手順で行う．具体的なブロック構成図を図 5.2 に示し，処理手順を図 5.3 に示す.

1) 環境コンテンツの読込み．必要があれば仮想 3 次元空間の設定.
2) 人体モデルの読込み．必要があれば保存されている歩行データの読込みや歩行状態などの設定を行う.
3) 歩行パスの設定.
4) 歩行動作生成に必要なパラメータの計算.
5) 歩行動作の生成.
   5.1) 移動速度などの入力.
   5.2) 運動方程式の計算.



(a) (b) (c)

図 5.4: 人物像モデル例.



(a) (b) (c)

図 5.5: Bezier 曲面を用いた人物像モデル例. 数字は Bezier パッチ数を表す.

   5.3) 一歩の歩行動作の生成.
   5.4) フレーム毎に表示.
6) 歩行データ動作や画像ファイルへの出力.

5.2.3 人物像モデル

ここでは，本研究で用いる人物像モデルの例を紹介する．人物像モデルは，図 3.1の構成に従い，人物腰部を原点とした階層構造をなしている.

図 5.4 は単純な図形で表した人物像モデルの例である．これらは，各部位がローカルな座標系を有する独立したオブジェクトとして構成されており，各部位を接続している関節ごとに，親オブジェクトに対する相対的な関節角度とスタック構造によって表示される．このような座標系の表し方の具体的なアルゴリズムおよび詳細は，文献 [115] を参照のこと.

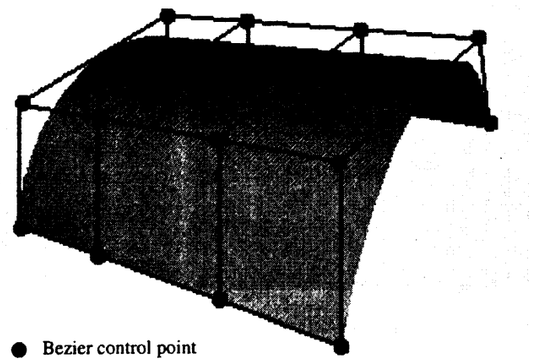


図 5.6: Bezier 曲面.

図 5.4 のモデルに，Bezier 曲面 [30] を適用して手足や衣服を生成させたものが図 5.5の人物像である．Bezier 曲面は，図 5.6 に示すように，いくつかの制御点を用いて曲面パッチを生成するもので，頂点指定によるポリゴンモデルに比べ，比較的少ない制御点の指定により，滑らかな曲面を生成することができる [42].

人物像に衣類を装着させる場合，衣類の動きの表現はもちろんのこと，衣類と人物像との相互作用も重要である [9]．Bezier 曲面を用いた場合には，少ない制御点で表現できると言う利点があるが，衣類と人体表面との交差判定が困難であるという問題があり，Bezier パッチを分割して交差判定を行う手法も試みられている [42].

第 4 章や第 5.4 節では，より複雑なポリゴンモデルを用いる例も紹介しており，例えば図 4.15や図 5.20では，VRML による人物像の 3 次元 CG モデルを用いている．また，第 5.6 節では，写真画像を用いる例についても言及する．

5.2.4 3 次元室内シーンへの適用例

図 5.7はこの典型的な応用例である．まず図 5.7 (a) において，歩行経路を指定する．この図でハイライト表示している部分が，マウスで指定した部分であり，歩行パスはこの指定した箇所を順に連結することで生成する．この連結した線分に対し，エリア分割及びステップ分割（第 3章）を行い，幾何的なパラメータを有するステップ・オブジェクトを生成する．次に図 5.7 (b) において，人物像モデルに対する様々な指定を行う．人物像モデルの選択，身長や体重，歩幅や水平移動速度，誇張表現を行うか否か，行うならばそのパラメータ，などである．図 5.7 (c) はアニメーションの生成結果例である．

同様に，図 5.8 も同じ室内シーンに対し，パス設定から歩行動作の生成までを行ったものであり，図 5.8 (e) では生成した複数の歩行者を表示している．

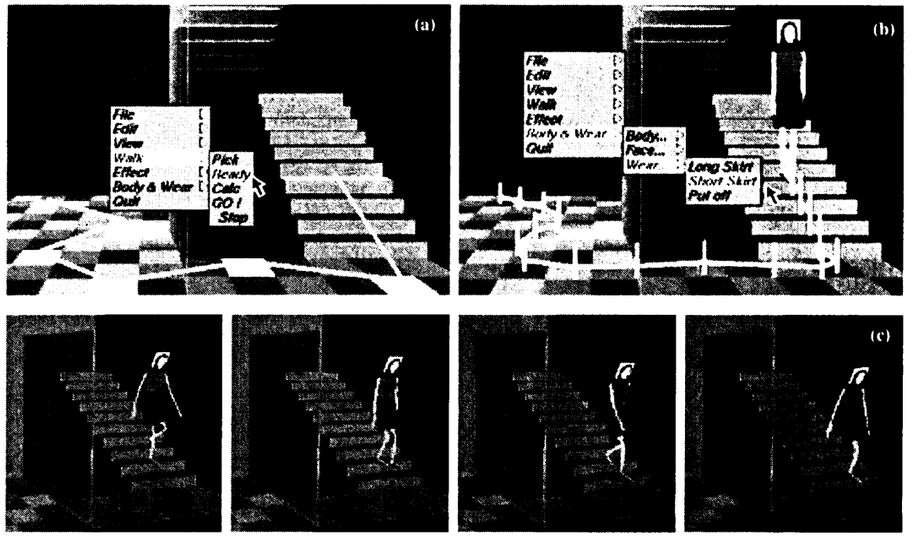


図 5.7: *WorldWideWalk* による生成例; (a) パス設定, (b) パラメータ計算, (c) 生成結果.

この例のように，室内シーンに歩行する人物像を容易に加えたり，建築物のシミュレーションを行うことが可能である．また，3 次元 CG であることから，容易に視点の変更を行うことができ，後述の図 5.14 (b) に示すように，ある歩行者の視点から見たシーン，いわゆるウォークスルーアニメーションを表示することも可能である．

図 5.9も，本システムを用いて実際にアニメーションを制作した例である．図 5.9は NTT サイバーソリューション研究所による研究技術紹介用ビデオパッケージ「梅美月」の 1 シーンであり，牛車の横の歩行者に対し，本論文で述べた歩行動作生成手法が適用されている．このようなシーンにおいて歩行動作を作成するには，熟練した CG アニメータでも数日を要するが，本システムではパスの指定を含めて数分で生成可能であり，産業的にも貢献が大きい．

5.3 3 次元環境でのパスの自動設定

この節では，複数人の歩行動作の生成を可能とするための，環境オブジェクトに対する歩行パスの設定方法について述べる．ここでの目的は，第 4 章で述べたような，多数の歩行者の運動更新や表示方法ではなく，多人数の歩行者のパスを設定する作業量を削減するための手法である．

一般に，3 次元仮想空間において複数の運動オブジェクトを生成する際に，各オブジェクト全てに運動を設定すると，大きな作業量を伴う．このような問題に対し，Reynolds の boid モデル [82] を用いた鳥や魚の群の生成を始めとして，鵜沼らによる群衆の歩行モデリング [109] などが報告されている．これらは，群あるいは群衆を支配する指向性と，他のオブジェクトとの衝突回避の方法

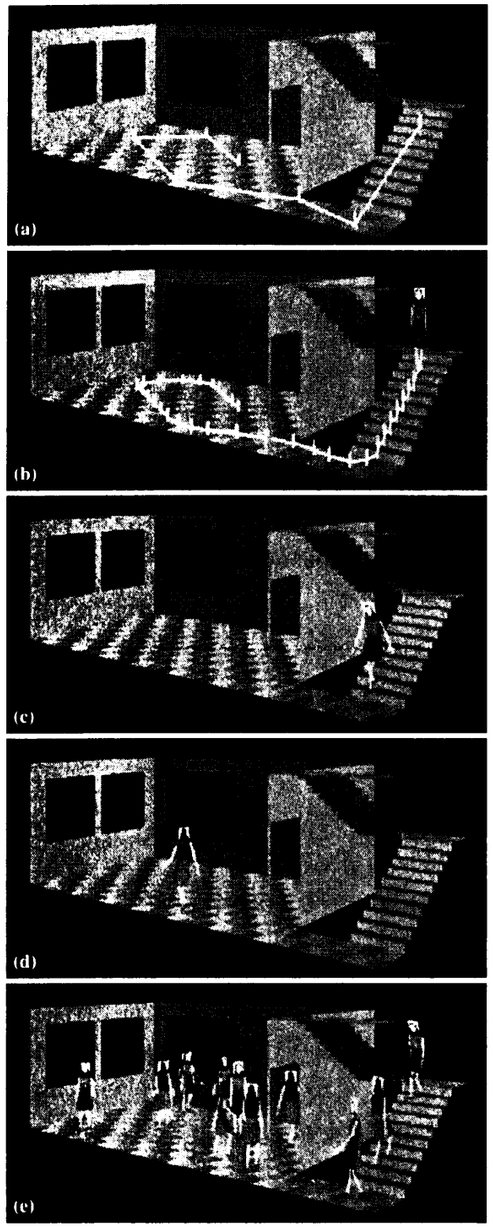


図 5.8: 室内シーンでの生成例: (a) パス設定, (b) ステップへの分割, (c) 階段下降, (d) 室内歩行, (e) 複数の歩行者生成.

図 5.9: ビデオ作品「梅美月」での歩行シーン.

とを記述することにより多数のオブジェクトの運動を効率良く生成できる手法を提供している．しかしながら，例えば建築物内の歩行などのように複雑な 3 次元環境モデルには適応していない．

一方，これまで述べてきた歩行動作生成システム（*WorldWideWalk*）においては，個別に歩行パスを設定しなければならないため，多数の人物像の歩行動作生成には適していない．従って，環境コンテンツに対し複数の人物像の歩行動作を効率良く生成するためには，個々の人物像に対する入力をできるだけ少なくすることが必要である．

この問題を解決するために，本節では以下の 2 点を考慮した：

1. 環境モデルを構成するオブジェクトに属性を付与することによる歩行経路の探索および設定．
2. 環境モデル内の障害物に対し反射方向を記述することによる方向転換の実現．

これにより，歩行開始点と初期の歩行方向のみを与えるだけで，方向転換を行いながら継続的に歩行する人物像の歩行動作を自動的に生成することが可能となった．

5.3.1 歩行経路の自動設定

本手法の特徴は，歩行開始点 P_0 と初期の歩行方向 $\boldsymbol{d}$ のみを与えるだけで，障壁で方向転換を行いながら歩行を継続することにある．本節では環境モデルを構成するオブジェクトに属性を付与することで，歩行経路の自動設定を実現する．

まず，環境モデルを構成するオブジェクトを歩行可能なオブジェクトと歩行不可能なオブジェクトとに分類する．歩行可能なオブジェクトとは人物像がその上を歩行することができるオブジェクトであり，具体的には床や階段を構成する面などを指す．

歩行不可能なオブジェクトは人物像が通過することができないオブジェクトであり，さらに動的な障害物と静的な障害物とに分類される．動的な障害物は他の人物像や可動物体であり，本節では考慮していない．静的な障害物は例えば壁面などのように不動のオブジェクトであり，本稿ではこれ以降「障壁」と記すことにする．

この障壁を通過することなく方向転換を繰り返すことにより，継続的な自動歩行が可能となる．従って，現在の歩行方向における障壁を見つけることと障壁における方向転換方法を記述することが本稿における目的となる．即ち，例えば図 5.10に示すように，与えられた環境モデルにおいて P_0 と $\boldsymbol{d}$ とが与えられた際に，歩行経路と最初の障壁との交点である P_1 を求める．

図 5.10において，P_0 を含み $\boldsymbol{d}$ に平行な鉛直平面を P_0 を通る鉛直線 c によって 2 分し，$\boldsymbol{d}$ の向きにある方の半平面を V とおく．V と交差する歩行可能なオブジェクトを $\boldsymbol{R} = \{\boldsymbol{R}_0, \boldsymbol{R}_1, \boldsymbol{R}_2, \ldots\}$ とおき，$\boldsymbol{R}$ と V との交差線分を $R = \{R_0, R_1, R_2, \ldots\}$ とおく．また，V と交差する障壁を $\boldsymbol{Q} = \{\boldsymbol{Q}_0, \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots\}$ とおき，$\boldsymbol{Q}$ と V との交差線分を $Q = \{Q_0, Q_1, Q_2, \ldots\}$ とおく．ただし，$\boldsymbol{R}_0$ は P_0 を含むオブジェクトである．このとき，P_1 を求める基本的な手順は以下の通りである:

1) $\boldsymbol{R}_0$ の V との交差線分 R_0 を計算し，$i = 0$ とする．

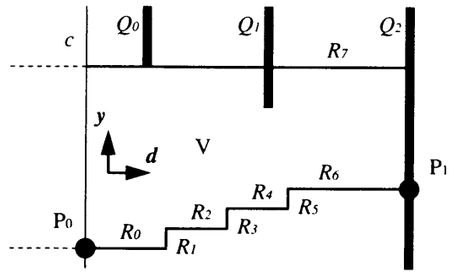


図 5.10: 障壁オブジェクトの概念．

2) $\boldsymbol{R}_i$ と隣接しているオブジェクトと V との交差線分のうち，R_i と幾何的に接続されているものを探索する．複数存在する場合は，$\boldsymbol{d}$ 方向にあり，R_i との距離が最も小さいものとする．
3) それが Q であれば求める障壁であり，探索を終了する．歩行可能なオブジェクトであれば R_{i+1} とし，その交差線分を含むオブジェクトを $\boldsymbol{R}_{i+1}$ とし，次のステップへ進む．
4) $i = i + 1$ とし，2) へ戻る．

なお，2 つのオブジェクトが隣接しているとは，{ **辺を共有している，交差している，辺で接している，点で接している，ある一定の距離以内にある** } のいずれかである．

以上の手順によって P_1 を求め，P_0 から P_1 に至る交差線分 R を連結することにより，直線的な単位歩行経路を決定することができる．このような単位歩行経路を連結していくことにより，継続的な歩行動作の生成が可能となる．

以上のような歩行経路の設定において不自然さが生じる場合，例えば V がオブジェクトの境界上に存在する場合や，人体の幅よりも狭い隙間に歩行経路が生成されるような場合は，環境オブジェクトに対して何らかの不可的な拘束条件を課する必要がある．

5.3.2 障壁における方向転換

次に，障壁においてどのように方向転換を行うかを記述する必要がある．実際は図 5.11に示すように，P_1 よりも幾分手前に方向転換を行う「折り返し点」P_1' を設置し，P_1' において次の歩行方向 $\boldsymbol{d}'$ を決定する．

このとき，最も簡単な方法は，例えば図 5.12に示す鏡面反射のように，入射角 θ_{in} の値によって反射角 θ_{out} を一意的に決定する方法であるが，初期位置 P_0 と初期方向 $\boldsymbol{d}$ によって全ての経路が決定されてしまう．このため，例えば図 5.13に示すように，乱数などを用いて適度な範囲で値を分散させることにより，適当な無秩序さを設けることができる．

次節で示す例では，障壁への入射角度と反射角度を

$$\theta_{out} = rand * \alpha + \theta_0,$$

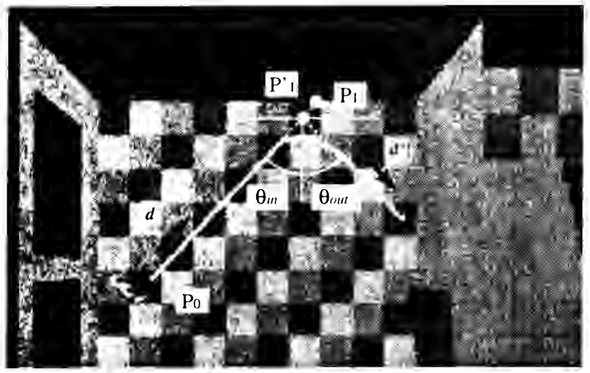


図 5.11: 壁面での反射モデル.

で表した. ここで, θ_0 は入射角度から求められる値であり, α は適度な振幅, $rand$ は $-1 \leq rand \leq 1$ であるような乱数値である.

さらに, 障壁の配置によって反射角度に束縛条件を適用することにより, ある程度の指向性を持たせることも考えられる. 例えば, 他の障壁に近い部分ではなるべく部屋の中央部へ方向転換するように反射角度を調節する, などである.

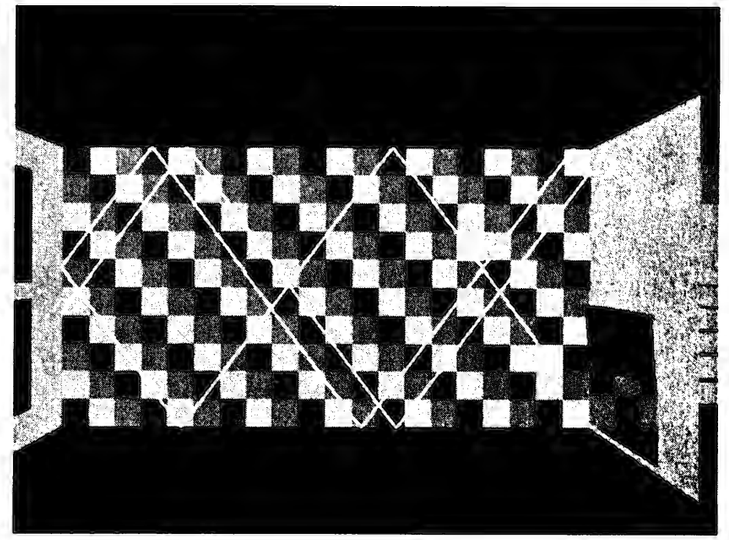
以上のように, 単位歩行経路を折り返し点で連結し, 図 5.11 に示すように徐々に方向転換を行うようにパスを近似することにより自然な歩行動作の生成が可能となる.

5.3.3 生成例と検討

図 5.14 (a) は, 本手法を 3 次元モデルとして定義された室内シーンに適用して生成した CG アニメーション例である. 4 人の人物像がそれぞれの歩行経路に沿って歩行している様子がわかる. 図 5.14 (b) はそのウォークスルー・アニメーションの 1 シーンであり, 図 5.14 (a) の階段を降りてきた人物の視点から見たものである. このように, 複数の人物像が同時に歩行するような場合でも各人の歩行動作を個別に生成して編集する必要がなくなり, 例えば建築物内を移動する, 背景としての多人数の歩行動作生成に対する本手法の有効性が伺える.

反面, 障壁に至るまでの経路が直線であり蛇行できない, といった点や, 各人の行動が環境オブジェクトのみに支配されているため, 個性ある行動の記述ができないといった問題点がある. また, 本稿では考慮していない他の動的なオブジェクトとの回避を採り入れた際に, その回避の方法によって前節で述べた歩行経路の計算を逐次修正しなければならない, といった問題点が予想され, 今後検討する余地がある.

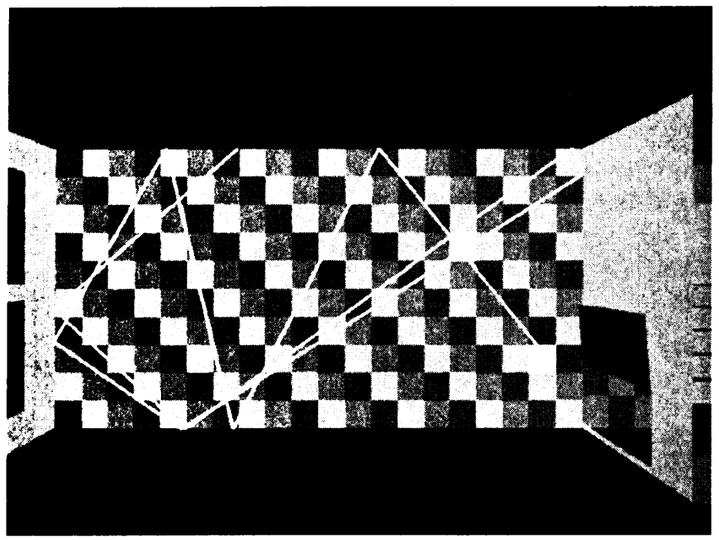
しかし, 初期位置と初期方向とを例えば乱数を用いて発生させることにより, 環境の一部としての動く人物像を大量に生成することが可能であり, コンテンツ生産に対するコストの削減に極めて有効である.



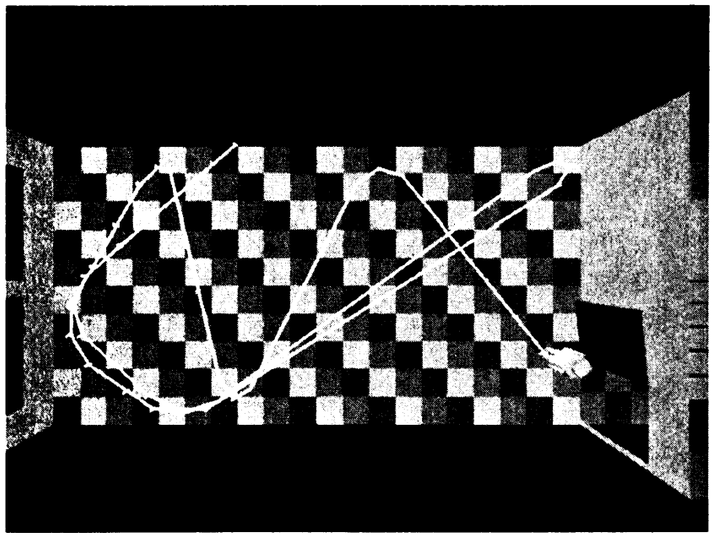
(a)

(b)

図 5.12: 鏡面反射モデル; (a) パス生成; ハイライト部分が初期位置と方向を指定する部分, (b) ステップへの分割.

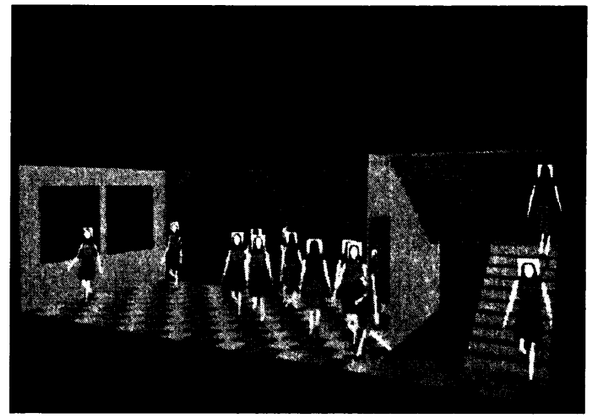


(a)

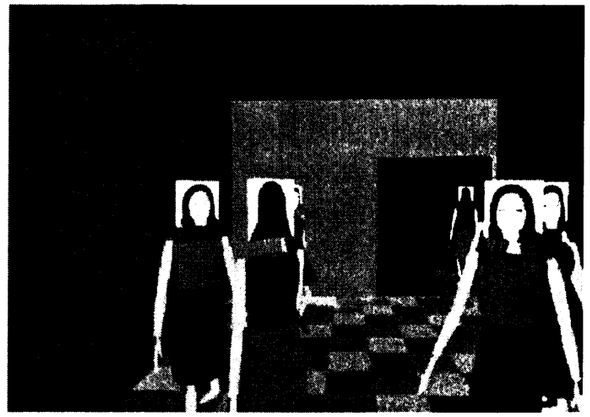


(b)

図 5.13: ランダム反射モデル; (a) パス生成; ハイライト部分が初期位置と方向を指定する部分, (b) ステップへの分割.



(a)



(b)

図 5.14: 複数歩行者の生成例; (a) 鳥瞰図, (b) 歩行者の視点.

5.4 2 次元背景画像との合成

この節では，奥行き情報を持たない 2 次元画像に対して歩行領域を設定し，3 次元歩行動作を行う人物像モデルを合成する手法について述べる [103][104]．本論文でこれまで述べた手法では，あらかじめ与えられている 3 次元 CG モデルによる環境を想定していた．しかし，3 次元環境モデルの作成は容易ではなく，現実的な環境モデルを作成するには，幾何的な形状の正確さと，色情報の両方が必要であり，一般にはテクスチャ・マッピングが用いられる [30][115] が，環境モデルを構成する全てのポリゴンを貼るためのテクスチャ画像を取得するのは容易ではない．

この問題を解決するために，1 枚の画像中のオブジェクトに擬似的な奥行情報を付与する手法 [45] や，画像撮影時のカメラパラメータを対話的に取得し，写真画像から可視である部分のテクスチャを取得して，3 次元形状を作成していく手法 [59][60] が報告されている．

本節で述べる手法は，

(1) 画像撮影時のカメラパラメータを対話的に設定し，

(2) そのカメラパラメータから得られる視点から見た 3 次元の歩行領域を画像中に設定し，

(3) 設定した歩行領域上に歩行者を生成する，

というものである．この手法は，得られた視点位置の変更を行わないことを前提としているが，画像からオブジェクトなどを抽出する必要がないため，背景画像をそのまま用いることができ，極めて容易に，現実味のある合成画像を得ることができる．

5.4.1 歩行領域の設定

2 次元画像は奥行情報を持たないため，歩行領域を 3 次元平面として設定し，画像撮影時（あるいは描画時）のカメラ情報を復元することによって仮想的な奥行情報を付与することで 3 次元歩行パスの設定を可能とする．

図 5.15 はカメラ情報の取得の原理を示している．ピンホールカメラを仮定し，さらに図 5.15 の 2 次元スクリーン上の矩形領域が 3 次元シーン内の水平面の長方形領域を射影したものであると仮定すると，平面の 3 次元位置情報，即ち図 5.15 の「3D plane」の縦横の辺の長さを与えれば，画像の 1 画素が 3 次元シーンのどれだけの長さに相当するかが計算でき，射影幾何学上の計算によってカメラ位置や焦点距離などの情報を得ることができるため [84][60]，画像上に仮想的な 3 次元平面の射影を生成することができる．

図 5.16 も，同様に二点透視によって撮影された画像のカメラ情報の取得を示している．

このようにして，仮想的な歩行領域を設定して人物像モデルと合成する．その際の表示のアルゴリズムは以下の通りである：

各フレームごとに，

(1) 背景画像を表示する．

(2) 取得したカメラパラメータ及び設定した歩行領域の位置情報を用いて，3 次元のワールド座標

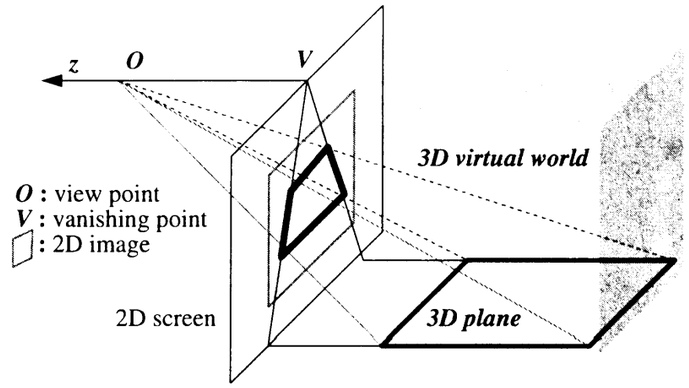


図 5.15: 一点透視図によって撮影された画像に対する，仮想的な奥行き情報の付与．

を生成し，視体積（viewing volume；[115] 参照のこと）を形成する．

(3) 歩行領域上に人物像モデルを生成し，(1) に対して上書き表示する．

即ち，背景画像はそのまま 2 次元画像として表示し，人物像は 3 次元 CG として得られたカメラパラメータを用いて表示することになる．

5.4.2 生成例と検討

図 5.17は一点透視で撮影された画像に対する奥行情報の設定と合成例を示す．図 5.17 (a) は歩行領域を設定しており，この領域は 3 次元の長方形領域を射影したものと仮定している．図 5.17 (b) はカメラパラメータ取得のために，歩行領域の位置情報を設定している．つまり，図 5.17 (a) の歩行領域の辺の（3 次元世界での）長さを指定している．この指定により，画像撮影時のカメラパラメータを取得することが可能となり，図 5.17 (c) に示すような歩行アニメーションとの合成が可能となる．なお，図 5.17 (c) においては，より現実感を出すために，人物像の足下に影を表すテクスチャ・マップされたポリゴンを表示している．

図 5.18は広角カメラで撮影された画像に対して人物像を合成した例である．図 5.19は，写真画像ではなくコンピュータ・グラフィクスで生成された背景画像に対して合成した例である．

このように，本節で述べた手法を用いることによって，複雑な環境モデルを作成することなく，現実味のある CG アニメーションを作成することが可能となる．

本手法は，背景画像から内部オブジェクトを抽出しないため，人物像の合成は背景画像に対する「上書き」しか実現していない．このため，例えば本来人物像よりも手前に描画されるべきオブジェクトが，人物像よりも奥の方に描画されてしまい，不自然な動画像を生成してしまうこともある．また，背景画像の内容を変化させないため，一度視点情報を取得した後に視点の移動を原則として行わない．このため，3 次元の人物像の動きを，常に一定の視点から見ることになるため，3 次元

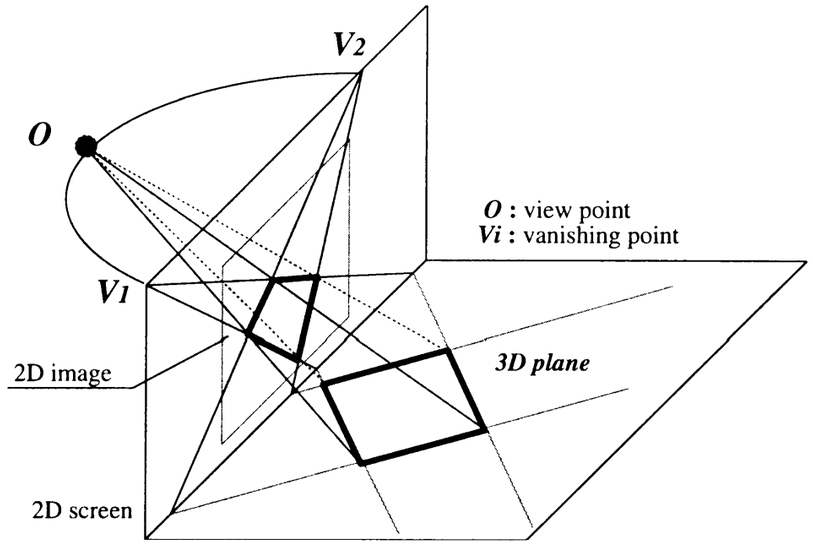


図 5.16: 二点透視図によって撮影された画像に対する，仮想的な奥行き情報の付与.

CG の有効性が半減されてしまうおそれもある.

しかしながら，画像という比較的入手しやすいコンテンツを用いて，現実味のある背景を有する 3 次元コンピュータ・アニメーションを容易に合成することができるという大きな特徴を持つため，例えば Web コンテンツや，ビデオ画像に対する特殊効果として有効に利用することが期待できる.

なお，画像の各画素が奥行情報を有するのであれば，前述の前後関係の問題は生ずることなく，より自然な合成シーンを得ることができる [104]. 例えば，Shinya らは，平行移動するカメラによって撮影されたビデオ画像から，エピポーラ処理によって画素の奥行情報を復元する手法を報告し [87]，南田らはその手法を基にして撮影時のカメラの手振れを補正する手法を報告している [63][64].

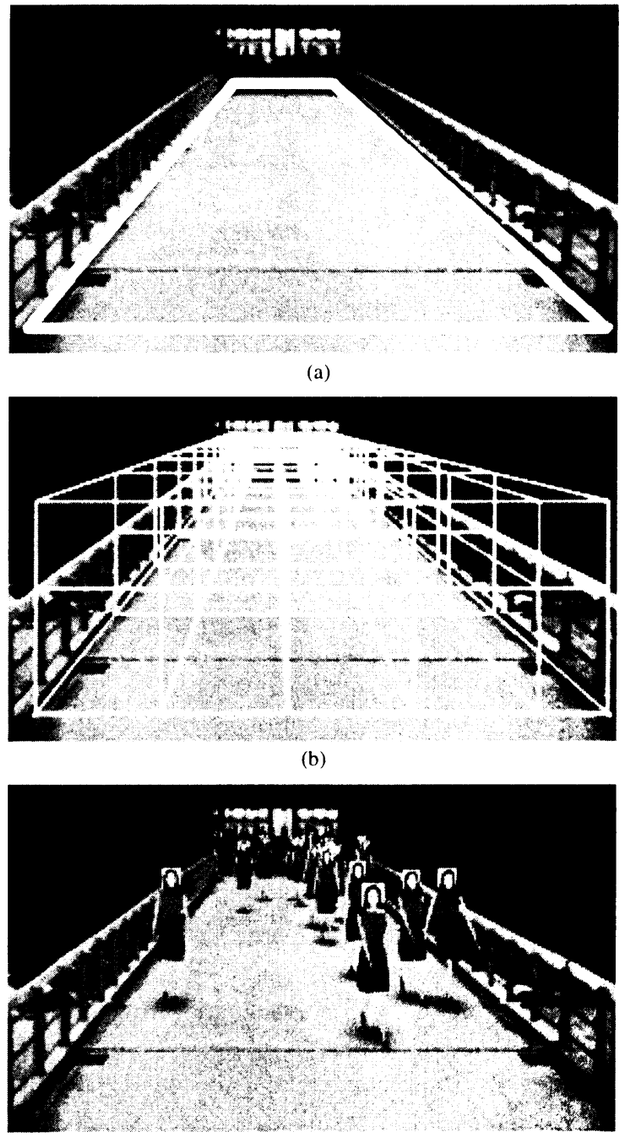
図 5.20および図 5.21は，そのようにして生成された奥行情報を持つ画像と，本手法でのを合成したシーンである.

近年は画像撮影時に奥行情報も同時に撮影可能なカメラ装置も現れており，ビデオ映像とコンピュータ・グラフィックスの融合はさらに発展するものと期待できる.

5.5 ネットワークを介した生成例

この節では，3 次元コンピュータ・アニメーションをネットワークを介した協調作業により生成・編集する手法について述べ，人物像歩行動作生成に適用した例を示す [102].

複数のユーザによる仮想環境の利用や仮想共有空間における協調作業については，様々な手法や

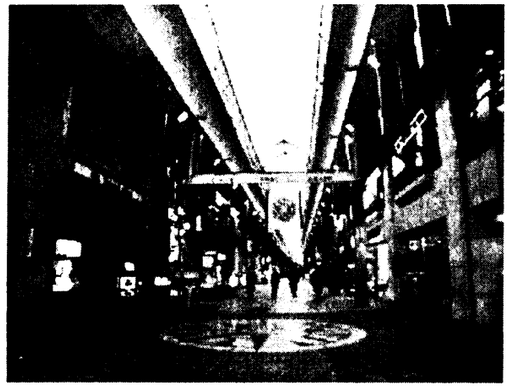


(a)

(b)

(c)

図 5.17: 歩行領域とカメラパラメータの設定例; (a) 歩行領域の設定, (b) サイズと縦横比の設定, (c) 合成例.

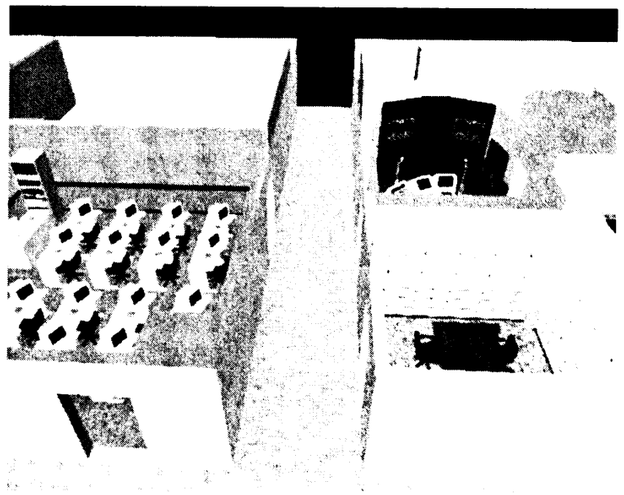


(a)

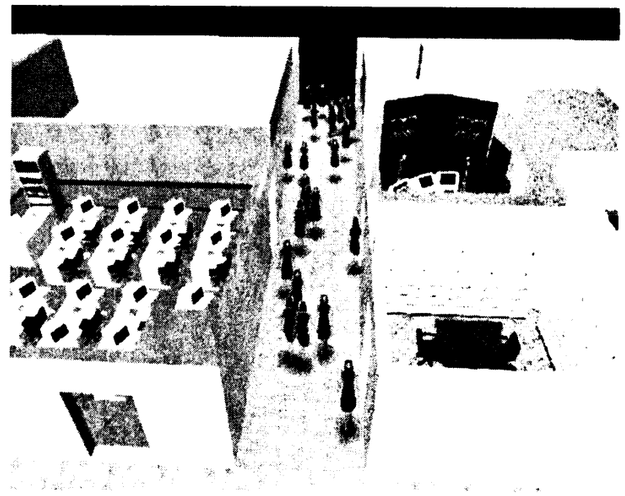
(b)

(c)

図 5.18: 2 次元画像との合成例 (1): (a) 背景画像, (b) スケルトンモデルを用いた合成例, (c) ポリゴンモデルを用いた合成例.



(a)



(b)

図 5.19: 2 次元画像との合成例 (2); (a) 背景 CG 画像, (b) 合成例.

図 5.20: VMODELER による出力画像との合成例 (1).

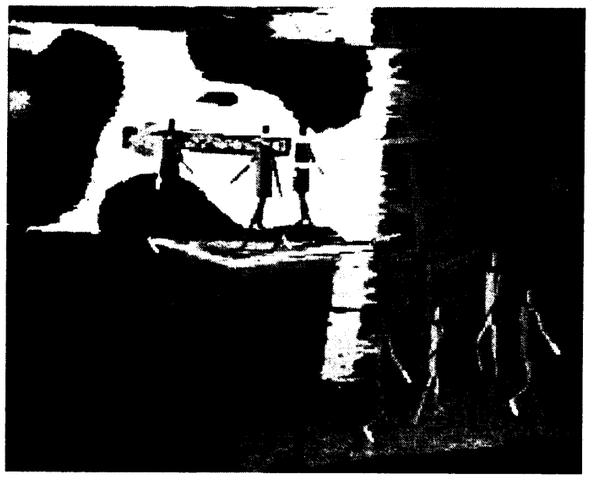


図 5.21: VMODELER による出力画像との合成例 (2).

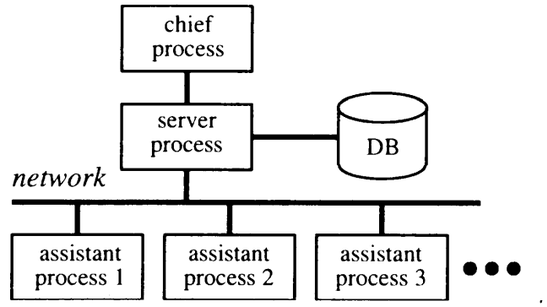


図 5.22: ネットワーク・モデル.

応用例が報告されている．そこでは各ユーザによるオペレーションを共有空間へ反映する際の伝達速度や同期処理などが主な問題となる．

一方，映像コンテンツの制作などにおいては仮想共有空間内のデータを変更してしまうことが多く，あるユーザによるあるオペレーションが望ましくない結果を生じることがあるため，適宜オペレーションを選択する必要がある．また，編集者が遠隔地にいる場合，時差も含めて同期処理が取りにくいため，一連の処理をまとめて非対話的に転送する方が効率が良い場合がある．このため，本手法では以下に示すような点を導入し，人物像歩行動作の協調作業による生成に適用した：

(1) 通常のサーバ & クライアント型モデルに特権クライアントを導入する，
(2) 特権クライアントプロセスにおいて，他のクライアントのオペレーションに対する選択機能を導入する，
(3) 各クライアントプロセスの共通リソースは各クライアントプロセスそれぞれが所持する，
(4) プロセス間の通信は，各データに対するオペレーションのみをサーバを介して転送し，必要に応じてシーンデータを転送する．

以上により，3 次元 CG アニメーションの協調作業環境による生成および編集を効率的に行う．

5.5.1 プロセスモデル

この節で採用しているネットワークモデルを図 5.22に示す．本モデルはサーバプロセス（SP），特権クライアントプロセスであるチーフプロセス（CP），通常のクライアントプロセスであるアシスタントプロセス（AP）の 3 種類のプロセスから構成される．

各クライアントは制作するシーンの初期データやアニメーション生成エンジンなどの共通リソースをサーバを介して得，各クライアント毎に編集を行う．CP は AP の編集操作を読み込むことでデータの更新を行う．コンテンツデータに対するオペレーションには，追加・修正や削除といったデータ構造に変化を加えるものも存在するため，各 AP の編集処理を全て受け入れるばかりではなく，適宜オペレーションを選択することが必要である．

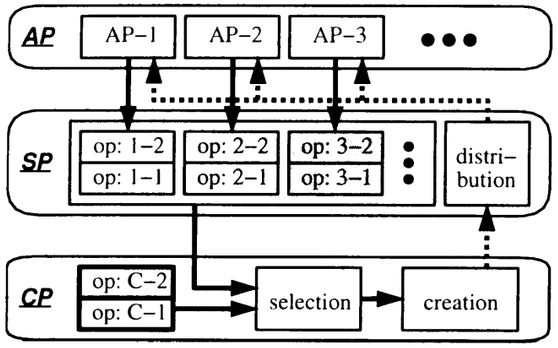


図 5.23: プロセス・モデル.

このため，図 5.23 に示すように，オペレーションを記述するデータがサーバに蓄積され，CP は SP を介して各 AP のオペレーションを得るが，その際に CP 自身によるオペレーションも含めて試行錯誤のうえ取捨選択し，シーンの生成を実行する．シーン生成後は決定された選択されたオペレーション列，あるいはシーンそのものを CP から SP へ転送し，サーバ上へ蓄積したり必要に応じて AP へ転送する．

以上の手順を繰り返すことにより，複数の AP による編集作業のうち，必要なオペレーションのみを選択して CP の上でシーンの生成を行うことが可能となる．

本論文で説明してきた歩行動作生成手法は，少数パラメータによる歩行動作の生成が容易に行えるため，あるクライアントで生成された歩行データを他のクライアントにフレーム毎に転送して表示させたり，あるいは歩行パスに関する情報のみを特定のクライアントに転送し，歩行動作の生成と表示を同時に実行させたりすることが可能である [102].

5.5.2 生成例

本手法を 3 次元モデルとして定義された室内シーンに適用して生成した CG アニメーションを図 5.24 に示す．この図は，はネットワークを介して歩行データを転送し，他のクライアントに表示させている例である．図 5.24 上段はそれぞれ AP により歩行動作を生成させながら歩行データを逐次的に転送している例を示し，図 5.24 下段は上段のプロセスからデータを受け取って表示させるとともに自プロセスでも歩行動作を生成している CP の例である．ただし，図 5.24 下段のプロセスは表示モデルが上段のプロセスに比べて複雑な人体モデルを使用していおり表示に時間がかかるため，位置の更新に遅延が見られる．これは，AP では操作を迅速に行うために単純な人物像モデルを利用し，一方，CP では映像チェックのため，より詳細な人物像モデルを利用していることを想定しているためである．これらの生成例から AP 並びに CP による協調編集作業が効率良く行われていることがわかる．

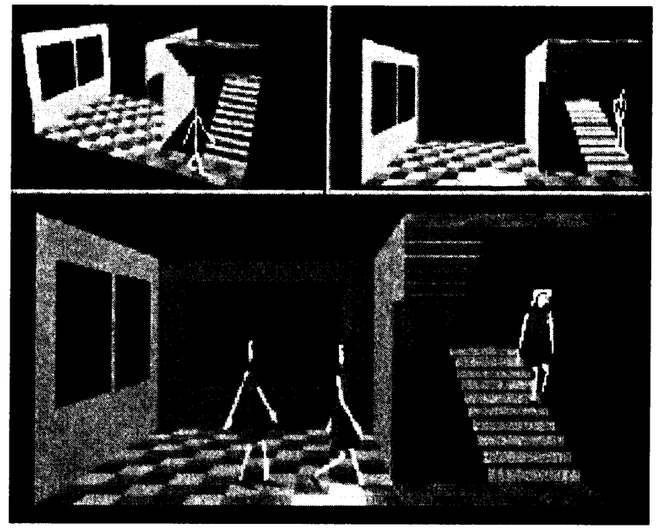


図 5.24: ネットワークを介したアニメーション生成例.

本手法で示したモデルでは，AP 間の相互作用や同期処理を制御することが困難であるという問題があるが，例えば顧客の要望に応じたシーンの生成などの協調作業を遠隔地にいる複数の編集者によって行うことが可能となる．また，オペレーションを蓄積することにより，実際のアニメーションシーンを保存するよりも記憶容量を必要としない，などの特徴がある．

5.6 画像を用いた人物像モデルを用いた生成

この節では，これまでの 3 次元の人物像モデルに代わる新しい人物像モデルの生成法に付いて述べる [106].

新谷，青木らは，2 次元画像から「階層構造をなすスケルトンと，スケルトンに対応づけられた画素」を有するオブジェクトを生成し，それに運動生成アルゴリズムを適用することで，静止画像を動画像化する手法を提案しており [2][1]，既に草木を対象としたアニメーション生成手法を提案している [88]．これは，画像の有する画素情報を人体を構成するボーンに対応づけてアニメーションを生成する van Overveld の手法 [72] を拡張したものである [89].

このように，特に写真画像の有する写実性を積極的に利用し，3 次元の動きや 3 次元 CG モデルと融合する手法が最近多く報告されている．例えば，Brand は顔の 3 次元モデルと写真とを合成し，顔画像が会話を行うアニメーションを生成している [16].

この節では，この手法を人物像全身モデルに適用し，画像から生成された 2 次元の人物像モデルに 3 次元の動作生成処理を行い，3 次元アニメーションを生成する手法について述べる．

本手法で行う処理は以下のようなものである：

1) 画像からの 2 次元人物像モデルの生成.

2) 人物像モデルに対する 3 次元動作生成エンジンの適用,

3) 3 次元アニメーションの表示.

これらの処理を行うためには，生成する人物像モデルに対し，

1. 人体の各部位の識別子及び階層構造,

2. 各部位の位置情報及び属性（相対位置，長さなど），

3. 各部位の形状および色情報,

の指定が必要である．本手法では，画像に対して入力された幾何的情報のみからこれらの情報を生成することで 2 次元人物像モデルの生成を容易化し，生成した人物像モデルを厚みのない 3 次元モデルとみなして動作生成エンジンを適用することで，3 次元アニメーションの生成を実現する．

5.6.1 人物像モデルの生成

幾何的情報の入力

図 5.25 (a) に示すように，人物像を含む画像において，人物像を構成する部位の端点及び関節点を入力する．スケルトンは，端点または関節点を結ぶ線分で表される．

人体構造及び属性の決定

上記の入力情報から，図 5.25 (b) に示すスケルトン情報を抽出し，この情報を基に各スケルトンの部位を識別し，さらに属性情報も付与する．

スケルトンの部位及び階層構造の決定は，スケルトンの有する接続数を用いて，あらかじめ設定されているテンプレートモデルの部位と各スケルトンとを一意的に対応づけることによって行う．左右対称な部位は，画像が正面像であるが背面像であるかを指定することで判別する．

各部位が　意的に対応づけられれば，各スケルトンの長さを部位の長さに対応づけられ，また，相対位置や初期姿勢を計算することが可能となる．なお，左右で長さが異るときは，長い方の値を用いる．

形状及びテクスチャ情報の取得

各スケルトンに対応づける画素情報を取得する．図 5.25 (a) (c) に示すように，スケルトン入力時にスケルトンに対応づける画素の範囲を表示し，対話的に修正し決定することで，スケルトンに対応づけるパーツ，即ちポリゴンとテクスチャを取得する．

5.6.2 アニメーション生成例

前節で述べた手法を用いて生成した 2 次元人物像モデルに対し，3 次元動作生成処理を適用し 3 次元表示を行う．歩行動作の生成は，スケルトンに対して本論文で述べて来た手法を用いて行う．図 5.26に実際に画像から人物像モデルを作成した例を示す．また，図 5.27 に多人数の生成例を示す．2 次元モデルを用いた場合，正面または背面以外の方向から見ると不自然さを生じてしまうと

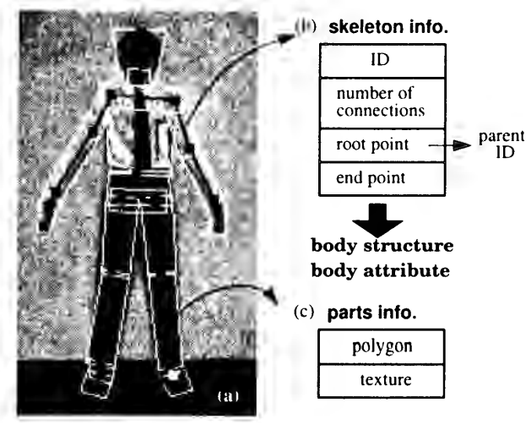


図 5.25: 画像とスケルトン情報，パーツ情報との対応.

いう問題があるが，歩行動作のように進行方向に対するねじれが少ない運動には，本手法で作成した人物像モデルで十分であり，本手法の有効性が伺える．

また，写真画像のみならず，CG キャラクタのように手足の長さの比率が人物像の標準的な比率とは異るような場合でも，本手法を適用することができる．

図 5.28は CG キャラクタの例であり，図 5.29，この CG キャラクタにスケルトンを設定して人物像モデルを生成し，背景画像中に歩行領域を設定して歩行アニメーションを生成した例である．この例からも分かる通り，本手法は 3 次元モデルを生成することなく，2 次元画像のみから 3 次元アニメーションを作成することが可能である．

本手法は，データ量の削減にも有効である．例えば図 4.15や図 5.20で用いている VRML による人物像の 3 次元 CG モデルは，ポリゴン数を比較的少なくしてあるものの，データ量が 1MB 以上ある．これに対して，図 5.25の例では，画像ファイル（JPEG 形式) が約 100KB，スケルトンおよびポリゴン情報が約 1KB であり，データ量を 1/10 に減らしている．よりデータ量の少ない画像ファイルを用いれば，よりデータ量を減ずることができる．さらに，モデル作成の作業時間についても，本手法は有利であり，上述の VRML モデルの作成には熟練した CG デザイナーが 1ヶ月以上を要しているが，図 5.26に示す作業は 15 分程度である．

本手法の特徴は，写真画像や CG 画像から容易に 2 次元人物像モデルを生成することが可能となることと，生成モデルへの 3 次元動作生成の適用により，現実味のある 3 次元アニメーションを生成可能なことである．本手法は，動物などの単純グラフ構造をなす構造体に有効であり，映像コンテンツの制作が容易になるものと期待できる．

なお，図 5.29のアニメーションは，NTT サイバーソリューション研究所による評価実験用アニメーションとして，実際にアニメータの手によって制作されたものである．また，本手法を実装したシステムは，平成 12 年 7 月 21 日から同年 8 月 6 日まで開催された「21 世紀夢の技術展」にお

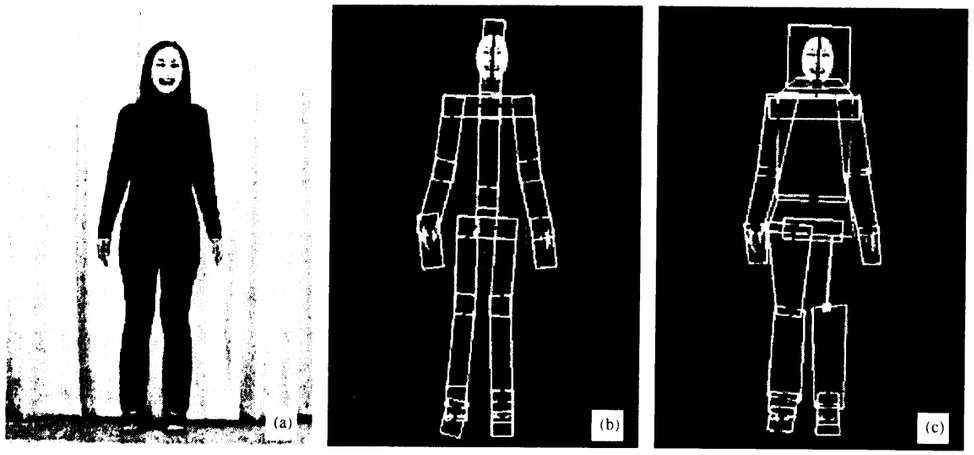


図 5.26: 画像を用いた人物像モデルによる歩行動作の生成例; (a) 原画像, (b) スケルトン情報の付与, (c) パーツ情報の付与, (d) スケルトンに対する歩行動作生成, (e) 生成例.

図 5.27: 画像を用いた人物像モデルによる多人数の歩行動作生成例.

いて展示され，好評を博した．図 5.30は，展示されたシステムによって制作されたアニメーションの例である．

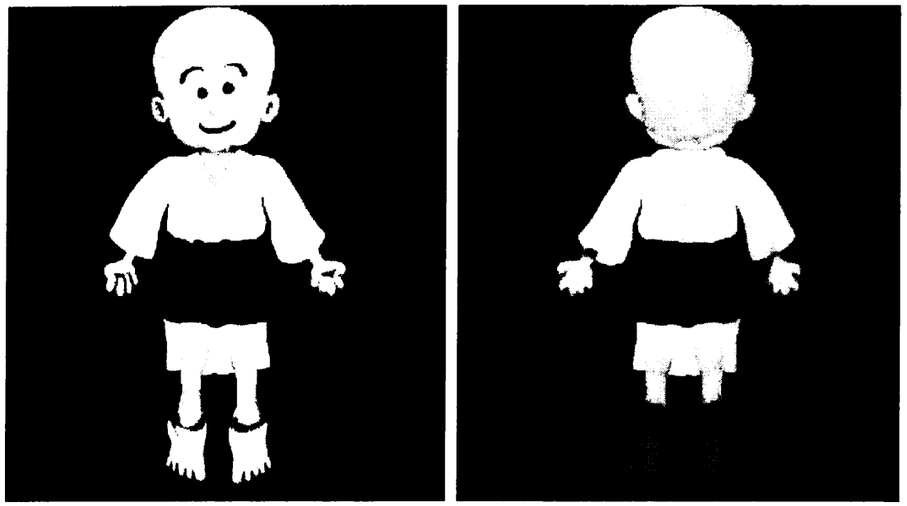
5.7 本章のまとめ

本章では，本研究で提案した歩行動作生成手法を，実際のアニメーション制作に適用するためにシステムとして実装したことについて述べ，さまざまな映像コンテンツに対する実現例を示し，映像制作現場における本研究の有効性を示した．

まず，仮想環境に対して与えられた歩行パスに沿う歩行動作を生成するためのアニメーション・システムの構成，処理手順について述べ，人物像モデルについて述べた．そして，具体的な 3 次元仮想環境への歩行パス設定と，そのパスに沿う歩行動作生成例について述べ，また，環境オブジェクトに対して属性や反射方向の記述モデルを付与することにより，歩行開始時の位置と方向を指定するだけで 3 次元環境モデルに適応する歩行動作を，継続して自動的に生成する手法について説明した．

次に，奥行情報を持たない 2 次元背景画像に対し，仮想的な 3 次元歩行領域を設定することによって奥行情報を設定し，同時にカメラパラメータを取得して背景画像と 3 次元人物像とを合成する手法について述べた．この成果により，容易に現実味のある映像シーンを制作することが可能となることを示した．

次に，ネットワークを介したアニメーションの協調制作についても述べた．本研究で提案した歩行動作生成手法は，少数パラメータで実現可能なため，ネットワークを介して通信するデータ量を減ずることが可能である．



(a) (b)

図 5.28: CG キャラクタ画像; (a) 正面図, (b) 背面図.

さらに，2 次元の人物画像を用いて，3 次元人物像モデルを作成し，3 次元歩行アニメーションを生成する手法について，そのアルゴリズムと実現例を示した．この成果によって，さらに現実味のあるアニメーション・シーンを制作することが可能となる．

このように，本章で述べた適用例の中には，実際に映像コンテンツ制作に使用されて公開されているものもあり，本アニメーション・システムの有効性と，産業界への貢献とを示している．

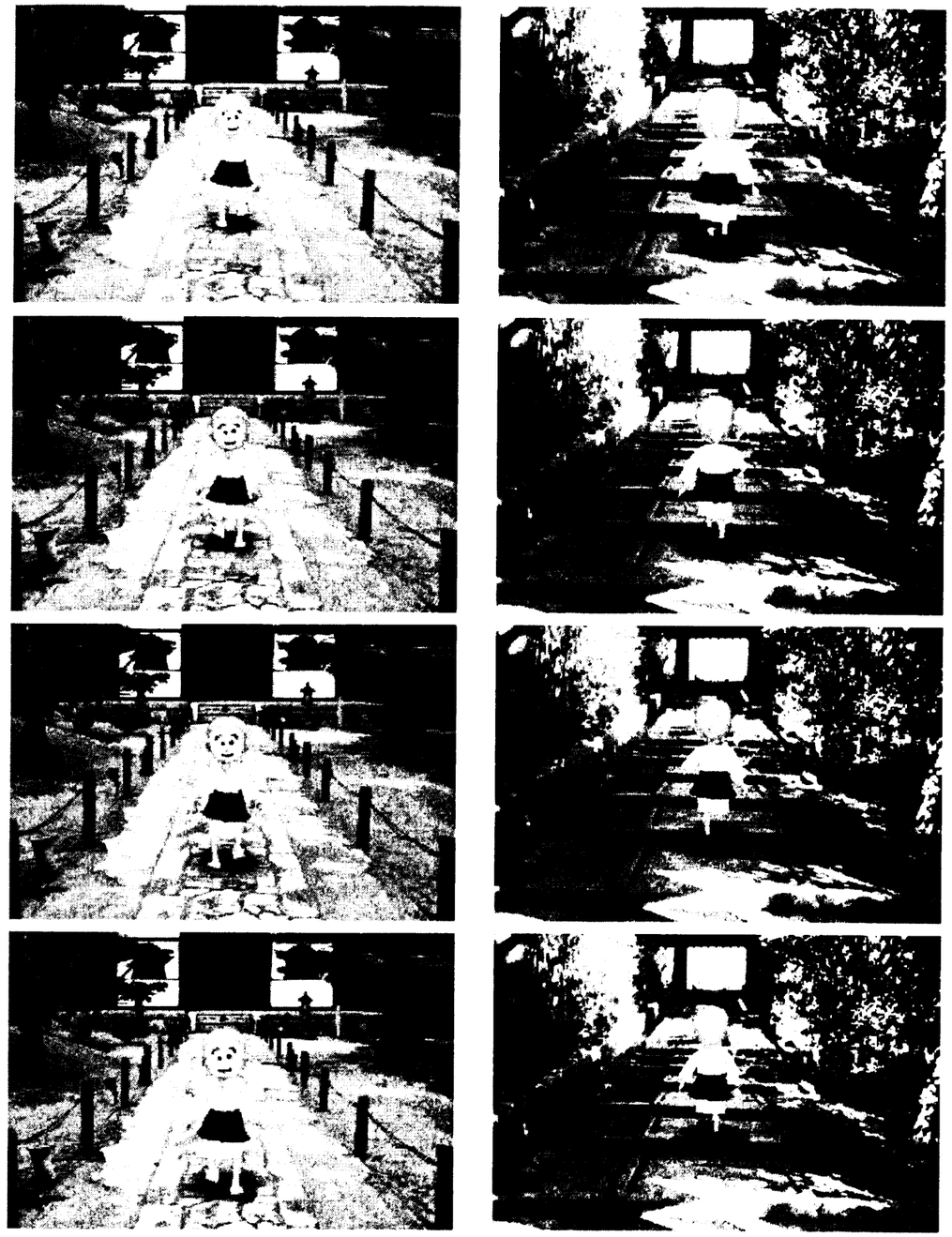


図 5.29: CG キャラクタを用いた歩行動作生成例.

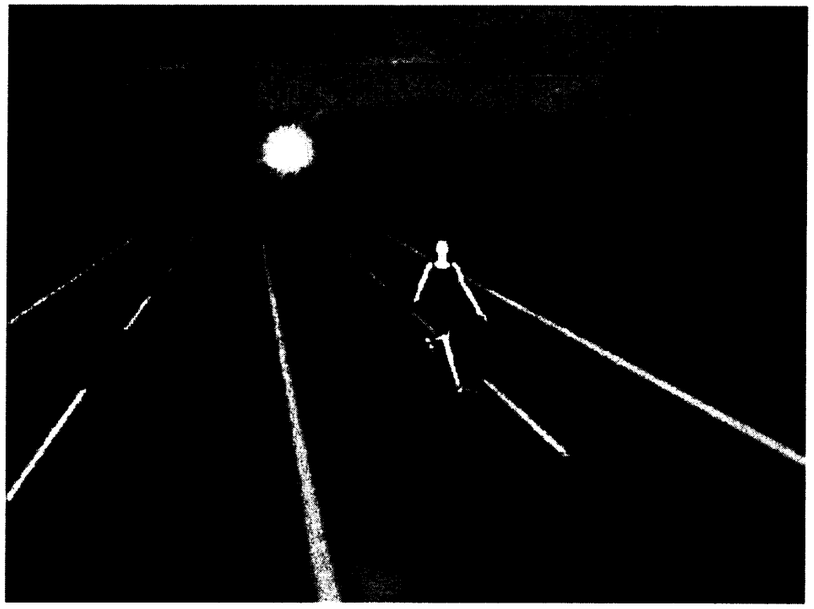


図 5.30: 「21 世紀 夢の技術展」に展示されたシステムで制作されたアニメーション例.

第 6 章

結論

本論文では，3 次元の多関節構造体で表された人物像の，3 次元仮想環境における歩行動作の生成方法に関する研究について述べた.

本研究における動作生成とは，以下のようなものであった：人物像を階層的な多関節構造体としてモデル化し，連結する剛体棒間の角度を関節角度とみなして，ある時刻における全関節角度をある条件のもとで計算する．その際，関節角度あるいは角速度の変化を，時間を変数として記述することにより，表示時刻単位で指定しなくても運動を自動的に生成し，表示することが可能となる.

しかしながら，人物のように多くの自由度を持つ複雑な多関節構造体の運動を時間を変数とする関数として記述することは困難であり，また，階段上昇・下降，左右への方向転換などを含む複雑な地形に合致する歩行動作を自動的に生成することには大きな困難が伴っていた.

本研究は，このような問題を解決するために，歩行動作を動力学と運動学を融合して生成する手法を採用し，あらかじめ入力された仮想的な 3 次元環境に対して移動経路を指定し，その経路上の歩行動作を自動的に生成するアプローチを提唱した．そして，それを実現するために，移動経路を分割して得られる一歩ごとのステップの列の，空間的な境界条件に合致するような歩行動作を生成する手法を研究し，報告した．さらに，そこで要求される，より少ない入力パラメータによる動作の自動生成と，より少ない計算量による効率的な生成とを実現した.

即ち，本研究の意義は，

- 平面内に限定されている歩行動作を 3 次元に拡張し，移動経路のみの設定で自動的に歩行動作を生成できるようにしたこと，
- 歩行動作に対する生成や制御を，より少ない入力で実現し，同時に，計算効率を向上させたこと，
- 実際にアニメーション・システムを構築し，さまざまな映像コンテンツの制作に適用したこと，

に集約される.

本論文においては，まず，第１章において本研究の概要を述べ，目的や内容の説明を行い，その意義を明確にした．

次に第２章において，コンピュータ・グラフィックスにおける人物像のアニメーションに関する技術動向や背景を述べ，本研究が人物像アニメーションの歴史的な流れの中でどのような位置付けにあり，どのような点を解決しようとしているのかを明確にした．即ち，本研究は，

- 動力学と運動学を融合して動作生成を行い，
- あらかじめ境界条件を設定して，その境界条件を満たすように動作を生成する，

という手法に分類され，任意の３次元地形に合致する歩行動作を自動的に，より少ないパラメータで生成し，制御することを可能とする手法を実現しなければならない，という点を明確にした．

第3章では，与えられた３次元仮想環境内に設定された任意の３次元移動経路に沿う歩行動作を，自動的に生成する手法について報告した．

ここでは，従来，動作範囲が平面上に限定されていたり，周囲の環境とは無関係である動きであった歩行動作の生成を，３次元環境に合致するように拡張すること，そして，従来，アニメーション制作に多大なコストを要していた歩行動作の設定を，できるだけ少ないパラメータの入力で行えるようにすること，を目的として，その解決をはかった．

第3章において，人物像の力学系を構築し，歩行動作を生成するための動力学変数を選定して，運動方程式を構築した．その際，歩行動作を構成する方程式を部分的な運動方程式に分離し，それぞれの運動を生成した後に合成する手法を提案した．そして，移動経路を一歩毎に分割する手法について述べ，一歩毎の歩幅と高低差を空間的な境界条件とし，また一歩に要する時間を時間的な境界条件とする歩行動作の動力学部分の運動方程式を立て，地形に適応した歩行動作を生成することを可能ならしめた．

まず，高低差ある地形形状に設定された歩行パスに対応した直進歩行動作を，自動的に生成する動作生成手法について述べ，生成例を示した．次に，人物像歩行動作に付随する，動力学モデルに基づく腕の運動の生成方法について述べた．この手法は，肘を曲げた状態にある腕全体をひとつの物理振子で近似した運動モデルを適用し，肘角度を運動学的に補間する手法を適用した．さらに，曲線歩行を含む，任意のパスに沿った人物像の歩行動作を自動的に生成する動作生成手法について詳細を述べ，その実現例と有効性を報告し，考察を行った．

これらの手法により，計算機による歩行動作生成の基本的な実装が可能となり，本研究の主要な目的である少数パラメータ入力による人物像の歩行動作生成が可能となった．また，複雑な地形を含む計算機内の３次元仮想環境内での，歩行動作生成の基本的な実装が可能となり，歩行パスの設定のみで複雑な地形変化に合致する歩行動作を生成することが可能となった．その結果，アニメーション制作者の歩行動作生成の負担が減少し，室内や各種建築物等の３次元環境モデルと歩く人物像とを融合することが可能となった．

続いて第４章では，個々の歩行動作に対する生成や制御，ならびに，多人数の歩行動作の生成や制御を，容易に，かつ少ない計算コストで実現する手法について報告した．

この章では，歩行動作生成の運動方程式において，直感的に把握しにくい外力の項を用いていることによる歩行状態の制御の困難性や，数値積分の際に生じる不安定性などの問題，そして，多数の人物像の運動を生成し表示する際の描画コストが増大の問題の解決をはかっている．

第４章において，まず，歩行動作の運動方程式における外力の項に床反力を近似した解析関数を用いる手法を提案し，運動方程式の外力の項に合理的な解釈を与えるとともに，逆問題を回避することによる計算の削減を実現した．次に，多数の人数の生成を目的とする際に，各人物像を運動の更新周期の異なる複数のグループに分類して生成する手法について，特にディジタルシティを始めとする仮想都市空間への適用を意識しながら報告した．

この研究成果により，個々の歩行者の歩行動作の計算を高速に，かつ，安定して行うことが可能となり，また，方程式の外力項を制御することにより，歩行状態の容易な制御を行うことが可能となった．また，多人数の運動生成ならびに表示に要する時間を，見た目の自然さを損なわずに軽減することが可能となり，実時間性を要求するアニメーション・システムへの適用を可能ならしめた．

第５章では，本研究成果を実際のコンテンツ制作に適用すべく，アニメーション・システムを構築し，様々なコンテンツ制作への適用した例について述べた．

まず，アニメーション・システムの構成や人物像について報告し，具体的な生成処理手順や生成例を示した．また，奥行情報を持たない２次元背景画像と３次元人物像との合成手法，ネットワークを介したアニメーションの協調制作，２次元の人物画像を用いた３次元人物像モデルの生成法についても報告した．その上で，実際の制作現場や産業展示会などでの利用例を紹介した．

第５章で述べたシステムにより，本論文で述べた研究成果を，実際の映像コンテンツ制作の場に提供することが可能となり，また，さまざまなシステムへ組み込むことが可能となる．

本研究により，計算機による人物像歩行動作の生成が可能となり，映像やコンピュータ・グラフィックスに関する産業界に従事する人々から自分自身の映像コンテンツを作りたい一般のユーザまで，さまざまな人たちに新しい形の映像製作ツールを提供できるものと期待できる．

本研究は，数多い人間の動作の中でも，歩行動作という限定された運動の生成のみを対象としたものであり，その意味で，計算機による人間の動作の生成という大きな目標に到達するには，遥かに遠い道程が残されている．現在の映像産業界は，現実世界からの様々なデータを取得して利用し，計算機による生成データと融合していく方向に進みつつある．それにも関わらず，アルゴリズムと計算機のみで人間の運動を生成することは，人間の創作意欲を駆り立ててやまないものがある．

本論文では，一貫して，いかに少ない入力で最大限の効果を上げるような歩行動作生成手法を提供できるか，という点を追求してきた．これは，現在のように計算機環境が整い，専門家でなくても，映像コンテンツやアニメーションを容易に制作することができる制作環境を提供するためである．ワードプロセッサの登場によって文章の生産量が飛躍的に増加したように（それによってもたらされた弊害も少なからず存在するが），映像，画像，その他のコンテンツの生産性を向上するこ

とが可能となるような，いわばワールド・プロセッサとも呼ぶべきツールを提供することができれば，これに過ぎる喜びはない．

謝辞

本研究をまとめるにあたり，有益な助言と懇切丁寧なご指導，並びに多大なるご尽力を頂きました石田 亨 京都大学教授に感謝致します．

本研究は，主として NTT ヒューマンインタフェース研究所， NTT サイバースペース研究所，NTT サイバーソリューション研究所において，多くの方々のご指導のもと行ったものです．

本研究をまとめるきっかけを与えてくださった曽根原 登 NTT サイバーソリューション研究所・研究プロジェクト・マネージャに感謝致します．本研究を始めるきっかけを作ってくださった 末永 康仁 名古屋大学教授，渡部 保日児 NTT サイバースペース研究所主幹研究員，本研究を進めるにあたり，様々な助言並びにご援助を下さった 徳永 幸生 芝浦工業大学教授，新谷 幹夫 NTT サイバーソリューション研究所主任研究員，高橋 時市郎 NTT サイバーソリューション研究所主幹研究員に感謝致します．また，本研究への技術的なアドバイスやご支援の言葉を頂きました Armin Bruderlin 氏に感謝致します．

個人名はとても書ききれませんが，日本電信電話株式会社旧ヒューマンインタフェース研究所・旧視覚情報研究部，旧マルチメディア処理研究部および旧映像処理研究部の皆様，サイバースペース研究所・旧メディア生成プロジェクトの皆様には，様々なご指導を頂きました．パーソナル・コンピュータもワークステーションもろくに使用できず，プログラミングの知識もなかった私がコンピュータ・グラフィックス関連の研究をまとめることができるようになったのも，ひとえに皆様のおかげです．厚く感謝致します．

また，サイバーソリューション研究所・コンテンツ流通プロジェクトの皆様，コミュニケーション科学基礎研究所・オープンラボの皆様，ディジタルシティ京都フォーラム会員の皆様， NTT ラーニングシステムズ映像制作事業部・制作技術部の皆様，常に人物像のモデルになって頂いた佐藤 美穂子嬢には，公私にわたり，非常にお世話になりました．ここに厚く御礼申し上げます．

さらに，旧ヒューマンインタフェース研究所の同期入社の皆さん，横須賀電気通信研究所のテニス部および週末テニスの仲間の皆さん，テニス合宿やスキー合宿でお世話になった皆さん，家族一同，そして妻の庄子，皆様の支援なくしては本研究をまとめることはできませんでした．

本研究を通じて，様々なコンピュータ・グラフィックスやアニメーションに関する知識を深めることができたとともに，人間の身体の複雑さ，すばらしさも改めて認識できました．

ここに感謝の意を表明し，厚く御礼申し上げます．

平成 12 年 11 月 筒口 けん

参考文献

[1] 青木 政勝, 新谷 幹夫, 筒口 拳, “動力学を利用した静止画の動画像化に関する検討”, 電子情報通信学会秋季ソサエティ大会（情報・システム）, pp.289 (D-12-67), September 1998.

[2] 青木 政勝, 新谷 幹夫, 筒口 拳, 曽根原 登, “静止画の動画像化における動力学パラメータの指定”, 電子情報通信学会春季総合大会（情報・システム）, pp.321 (D-12-148), March 1999.

[3] W. W. Armstrong, “Recursive Solution to the Equations of Motion of an N-link Manipulator”, *Proceedings of the Fifth World Congress on Theory of Machines and Mechanisms*, pp.1343–1346, 1979.

[4] W. W. Armstrong and M. Green, “Articulated Rigid Bodies for Purpose of Animation”, *Proceedings of Graphics Interface 1985*, pp.407–415, 1985.

[5] W. W. Armstrong, M. Green and R. Lake, “Near-Real-Time Control of Human Figure Models”, *IEEE Computer Graphics and Aplications*, **7**(6), pp.52–61, June 1987.

[6] N. I. Badler, K. H. Manoochehri and G. Walters, “Articulated Figure Positioning by Multiple Constraints”, *IEEE Computer Graphics and Aplications*, June 1987.

[7] N. I. Badler, B. A. Bersky and D. Zeltzer, *MAKING THEM MOVE: Mechanics, Control, and Animation of Articulated Figures*, Morgan Kaufmann Publishers, 1991.

[8] N. Badler, C. Phillips and B. Webber, *Simulating Humans: Computer Graphics, Animation, and Control*, Oxford Press, 1993.

[9] D. Baraff and A. Witkin, “Large steps in Cloth Simulation”, *Proceedings of SIGGRAPH 1998*, pp.43–54, 1998.

[10] A. H. Barr, “Teleological Modeling”, *SIGGRAPH 1989 Cource Note 30*, pp.B1–B7, 1989.

[11] R. Barzel and A. H. Barr, “Controlling Rigid Bodies with Dynamic Constraints”, *SIGGRAPH 1989 Cource Note 30*, 1989.

[12] R. Barzel. *Physically-Based Modeling for Computer Graphics*, Academic Press, INC., 1991.

[13] L. Bezault, R. Boulic, N. M. Thalmann and D. Thalmann, "An Interactive Tool for the Disign of Human Free-Walking Trajectories", *Creating and Animating the Virtual World*, Springer-Verlag, pp.87–104, 1992.

[14] R. Boulic, N. M. Thalmann and D. Thalmann, "A Grobal Human Walking Model with Real-time Kinematic Personification", *The Visual Computer*, **6**(6), pp.344–358, 1990.

[15] R. Boulic, Z. Huang, N. M. Thalmann and D. Thalmann, "Goal-oriented Design and Correction of Articulated Figure Motion with the Track System", *Computers & Graphics*, **18**(4), pp.443–452, 1994.

[16] M. Brand, "Voice Puppetry", *Proceedings of SIGGRAPH 1999*, pp.21–28, 1999.

[17] R. A. Brooks, "New Approaches to Robotics", *Science*, **253**, pp.Article 1227–1232, 1991.

[18] A. Bruderlin, "Goal-Directed, Dynamic Animation of Bipedal Locomotion", Master's Thesis, Simon Fraser University, 1988.

[19] A. Bruderlin and T. W. Calvert, "Goal-Directed, Dynamic Animation of Human Walking", *Computer Graphics*, **23**(3), pp.233–242, 1989.

[20] A. Bruderlin, "Dynamics + Kinematics = Controllable Realism", *SIGGRAPH 1990 Course Note*, pp.136–144, 1990.

[21] A. Bruderlin and T. Calvert, "Interactive Animation of Personalized Human Locomotion", *Proceedings of Graphics Interface 1993*, pp.17–23, 1993.

[22] A. Bruderlin, C. G. Teo and T. Calvert, "Procedural Movement for Articulated Figure Animation", *Computers & Graphics.*, **18**(4), pp.453–461, 1994.

[23] A. Bruderlin and L. Williams, "Motion Signal Processing", *Proceedings of SIGGRAPH 1995*, pp.97–104, 1995.

[24] T. W. Calvert, C. Welman, S. Gaudet, T. Schiphorst and C. Lee, "Composition of Multiple Figure Sequences for Dance and Animation", *The Visual Computer*, **7**, pp.114–121, 1991.

[25] T. Calvert, A. Bruderlin, J. Dill, T. Schiphorst and C. Welman, "Desktop Animation of Multiple Human Figures", *IEEE Computer Graphics & Applications*, pp.18–26, May 1991.

[26] D. A. Carlson and J. K. Hodgins, "Simulation Levels of Detail for Real-time Animation", *Proceedings of Graphics Interface 1997*, pp.1–8, 1997.

[27] J. Cohen, M. Olano and D. Manocha, "Appearance-Preserving Simplification", *Porceedings of SIGGRAPH 1998*, pp.115–122, 1998.

[28] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes", *Proceedings of SIGGRAPH 1995*, pp.173–182, 1995.

[29] T. Ertl, H. Ruder, R. Allrutz, K. Gruber, M. Ruder, J. Subke and K. Widmayer, "Interactive control of biomechanical animation", *The Visual Computer*, **9**, pp.459–465, 1993.

[30] J. D. Foley, A. van Dam, S. K. Feiner and J. H. Hughes, *Computer Graphics: Principles and Practice, 2nd Edition in C*, Addison-Wesley, 1996.

[31] K. S. Fu, R. C. Gonzalez and C. S. G. Lee, *ROBOTICS: Control, Sending, Vision, and Intelligence*, McGraw-Hill, 1987 (邦訳: 本多庸悟 監訳, ロボティクス, 日刊工業新聞社, 1989).

[32] 藤田 恒太郎, 寺田 春水, 生体観測, 南山堂, 1976.

[33] 福井 一夫, "コンピュータアニメーション", テレビジョン学会誌, **46**(9), pp.1141–1148, 1992.

[34] 福井 一夫, "多関節構造体の動作入力システム", *PIXEL*, **112**, pp.77–83, 1992.

[35] J. Funge, X. Tu and D. Terzopoulos, "Cognitive modeling: knowledge, reasoning and planning for intelligent characters", *Proceedings of SIGGRAPH 1999*, pp.29–38, 1999.

[36] T. A. Funkhouser and C. H. Séquin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization on Complex Virtual Environments", *Proceedings of SIGGRAPH 1993*, pp.247–254, 1993.

[37] M. Garland and P. S. Heckbert, "Surface Simplification Using Quadric Error Metrics", *Proceedings of SIGGRAPH 1997*, pp.209–216, 1997.

[38] M. Girard, "Interactive Design of 3D Computer-Animated Legged Animal Motion", *IEEE Computer Graphics & Applications*, **7**(6), pp.39–51, 1987.

[39] M. Gleicher, "Retargetting Motion to New Characters", *Proceedings of SIGGRAPH 1998*, pp.33–42, 1998.

[40] R. Grzeszczuk, Demetri Terzopoulos and Geoffrey Hinton, "NeuroAnimator: Fast Neural Network Emulation and Control of Physics-based Models", *Proceedings of SIGGRAPH 1998*, pp.9–20, 1998.

[41] B. Guenter, C. Grimm, D. Wood, H. Malvar and Fredric Pighin, "Making Faces", *Proceedings of SIGGRAPH 1998*, pp.55–66, 1998.

[42] 花里 高宏, 筒口 拳, 古川 貴雄, 曽根原 登, 清水 義雄, "双3次 Bezier パッチの適応的分割を用いた衣服形状生成", 情報処理学会「グラフィクスと *CAD*」研究報告, **97**(86), pp.61–66, August 1997.

[43] J. K. Hodgins, W. L. Wooten, D. C. Brogan and J. F. O'Brien, "Animating Human Athletics", *Proceedings of SIGGRAPH 1995*, pp.71–78, 1995.

[44] H. Hoppe, "Progressive Meshes", *Proceedings of SIGGRAPH 1996*, pp.99–108, 1996.

[45] Y. Horry, K. Anjyo and K. Arai, "Tour Into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image", *Proceedings of SIGGRAPH 1997*, pp.225–232, 1997.

[46] P. M. Isaacs and M. F. Cohen, "Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics", *Computer Graphics*, **21**(4), pp.215–224, 1987.

[47] T. Ishida, J. Akahani, K. Hiramatsu, K. Isbister, S. Lisowski, H. Nakanishi, M. Okamoto, Y. Miyazaki and K. Tsutsuguchi, "Digital City Kyoto: Towards A Social Information Infrastructure", M. Klusch, O. Shehory and G. Weiss (Eds.): *Cooperative Information Agents III*, LNAI 1652, Springer-Verlag, pp.23–35, 1999.

[48] T. Ishida and K. Isbister (Eds.), *Digital Cities: Technologies, Experiences, and Future Perspectives*, LNCS 1756, Springer-Verlag, 2000.

[49] Kahle, Leonhardt and Platzer, 解剖学アトラス, 越智淳三 訳, 文光堂, 1981.

[50] M. J. Kilgard, *Programming OpenGL for the X Window System*, Addison-Wesley, 1996.

[51] H. Ko and J. Cremer, "VRLOCO: Real-Time Human Locomotion from Positional Input Streams", *Presence*, **5**(4), pp.367–380, 1996.

[52] H. Ko and N. I. Badler, "Animation Human Locomotion with Inverse Dynamics", *IEEE Computer Graphics and Applications*, **16**(2), pp.50–59, 1996.

[53] Y. Koga, K. Kondo, J. Kuffner and J. C. Latombe, "Planning Motions with Intentions", *Proceedings of SIGGRAPH 1994*, pp.395–408, 1994.

[54] J. Laszlo, M. van de Panne and E. Fiume, "Limit Cycle Control and its Application to the Animation of Balancing and Walking", *Proceedings of SIGGRAPH 1996*, pp.155–162, 1996.

[55] J. Lee and S. Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for human-like figures", *Proceedings of SIGGRAPH 1999*, pp.39–48, 1999.

[56] P. Lee, S. Wei, J. Zhao and N. I. Badler, "Strength Guided Motion", *Computer Graphics*, **24**(4), pp.253–262, 1990.

[57] D. Luebke and C. Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments", *Proceedings of SIGGRAPH 1997*, pp.199–208, 1997.

[58] D. Manocha, T. Hudson and K. E. Hoff III, "Visibility Culling using Hierarchical Occlusion Maps", *Proceedings of SIGGRAPH 1997*, pp.77–88, 1997.

[59] 松田 達樹, 新谷 幹夫, 曽根原 登, "画像と投影拘束を利用した三次元形状モデリング", 電子情報通信学会秋季ソサイエティ大会（情報・システム）, pp.291(D-12-69), September, 1998.

[60] 松田 達樹, 新谷 幹夫, 曽根原 登, "投影拘束を利用した対話的モデリングシステムの構築", 電子情報通信学会春季総合大会（情報・システム）, pp.303(D-12-130), March 1999.

[61] M. McKenna and D. Zeltzer, "Dynamic Simulation of Autonomous Legged Locomotion", *Computer Graphics*, **24**(4), pp.29–38, 1990.

[62] M. McKenna and D. Zeltzer, "Dynamic Simulation of a Complex Human Figure Model with Low Level Behavior Control", *Presence*, **5**(4), pp.431–456, 1996.

[63] 南田 幸紀, 新谷 幹夫, 曽根原 登, "エピポーラ画像解析のための画像補正処理", 電子情報通信学会秋季ソサイエティ大会, pp.(D-12-71), September 1998.

[64] 南田 幸紀, 新谷 幹夫, 曽根原 登, "エピポーラ画像解析のための撮影時刻補正処理", 電子情報通信学会春季総合大会, pp.D-11-95, March 1999.

[65] C. T. Morgan, J. S. Cook III, A. Chapanis and M. W. Lund, 人間工学データブック, 近藤武 他訳, コロナ社, 1977.

[66] F. Multon, L. France, M. P. Cani-Gascuel and G. Debunne, "Computer Animation of Human Walking: a Survey", *The Journal of Visualization and Computer Animation*, **10**(1), pp.39–54, 1999.

[67] 中村隆一, 斎藤宏, 臨床運動学（第2版）, 医歯薬出版株式会社, 1990.

[68] 中村隆一, 斎藤宏, 基礎運動学（第4版）, 医歯薬出版株式会社, 1992.

[69] 人間工学人体計測編集委員会, 人体計測図表, 医歯薬出版株式会社, 1968.

[70] 人間工学ハンドブック編集委員会, 人間工学ハンドブック増補第2版, 金原出版株式会社, 1972.

[71] M. O'Rourke, *Principles of Three-Dimensional Computer Animation: Modeling, Rendering, and Animation with 3D Computer Graphics*, W. W. Norton & Company, 1995（邦訳: 袋谷賢吉・大久保篤志 共訳, 3次元コンピュータ・アニメーションの原理, トッパン, 1997）.

[72] C. W. A. M. Van Overveld, "A Technique for Motion Specification in Computer Animation", *The Visual Computer*, **6**, pp.106–116, 1990.

[73] C. W. A. M. Van Overveld and H. Ko, "Small Steps for Mankind: Toward a Kinematically Driven Dynamic Simulation of Curved Path Walking", *The Journal of Visualization and Computer Animation*, **5**, pp.143–165, 1994.

[74] M. van de Panne, E. Fiume and Z. G. Vranesic, "Control Techniques for Physically-Based Animation", *EUROGRAPHICS workshop on Animation: Eurographics Technical Report Series ISSN 1017-4656*, 1992.

[75] M. van de Panne, "From Footprints to Animation", *Computer Graphics Forum*, **16**(4), pp.211–223, 1997.

[76] K. Perlin, "Real Time Responsive Animation with Personality", *IEEE Transaction on visualization and computer graphics*, **1**(1), pp.5–15, 1995.

[77] K. Perlin and A. Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds", *Proceedings of SIGGRAPH 1996*, pp.205–216, 1996.

[78] C. B. Phillips and N. I. Badler, "Interactive Behaviors for Bipedal Articulated Figures", *Computer Graphics*, **25**(4), pp.359–362, July 1991.

[79] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski and D. H. Salesin, "Synthesizing Realistic Facial Expressions from Photographs", *Proceedings of SIGGRAPH 1998*, pp.75–84, 1998.

[80] Z. Popović and A. Witkin, "Physically Based Motion Transformation", *Proceedings of SIGGRAPH 1999*, pp.11–20, 1999.

[81] M. H. Raibert and J. K. Hodgins, "Animation of Dynamic Legged Locomotion", *Computer Graphics*, **25**(4), pp.349–358, 1991.

[82] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Computer Graphics*, **21**(4), pp.25–34, 1987.

[83] 臨床歩行分析懇談会, 臨床歩行分析入門, 医歯薬出版株式会社, 1989.

[84] D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics, 2nd Edition*, McGraw Hill, 1990.

[85] C. Rose, B. Guenter, B. Bodenheimer and M. F. Cohen, "Efficient Generation of Motion Transitions using Spacetime Constraints", *Proceedings of SIGGRAPH 1996*, pp.147–154, 1996.

[86] J. Rose and J. G. Gamble, *Human Walking Second Edition*, Williams & Wilkins, Baltimore, 1994.

[87] M. Shinya, T. Saito, T. Mori and N. Osumi, "VR Models from Epipolar Images: An Approach to Minimize Errors in Synthesized Images", *Computer Vision ACCV 1998*, LNCS 1352, **II**, pp.471–478, 1998.

[88] M. Shinya, M. Aoki, K. Tsutsuguchi and N. Konani, "Dynamic Texture: Physically Based 2D Animation", *Proceedings of SIGGRAPH 1999 Conference Abstracts and Applications*, pp.239, August 1999.

[89] 新谷 幹夫, 青木 政勝, "力学シミュレーションによる2次元アニメーション", 電子情報通信学会技術研究報告, **DSP99**(97), pp.67–73, October 1999.

[90] A. J. Stewart and J. F. cremer, "Beyond Keyframing: An Algorithmic Approach to Animation", *Proceedings of Graphics Interface 1992*, pp.273–281, 1992.

[91] G. Taga, Y. Yamaguchi and H. Shimizu, "Self-organized control of bipedal locomotion by nural oscillators in unpredictable environment", *Biological Cybernetics*, **65**, pp.147–159, 1991.

[92] 多賀厳太郎, “下肢運動のモデル化とシミュレーション”, バイオメカニズム学会誌, **16**, pp.209–214, 1992.

[93] N. M. Thalmann, R. Laperriere and D. Thalmann, “Joint-Dependent Local Deformations for Hand Animation and Object Grasping”, *Proceedings of Grphics Interface 1988*, pp.26–33, 1988.

[94] N. M. Thalmann and D. Thalmann, “Complex Models for Animationg Synthetic Actors”, *IEEE Computer Graphics & Applications*, **11**(5), pp.32–44, 1991.

[95] N. Torkos and M. van de Panne, “Footprint-based Quadruped Motion Synthesis”, *Proceedings of Graphics Interface 1998*, pp.151–160, 1998.

[96] R. Trappl and P. Petta (Eds.), *Creating Personalities for Synthetic Actors*, LNAI 1195, Springer, 1997.

[97] 筒口 拳, 渡部 保日児, “人物の歩行動作生成手法”, 「グラフィクスと *CAD*」 シンポジウム講演論文集, pp.189–197, 1993.

[98] K. Tsutsuguchi, H. Sakaino and Y. Watanabe, “Terrain Adaptive Human Walking”, *Insight Through Computer Graphics (Proceedings of CG International 1994)*, World Scientific, pp.180–191, 1996.

[99] 筒口 拳, 境野 英朋, 渡部 保日児, “地形適応型歩行動作生成手法を用いた人物の歩行アニメーション”, 電子情報通信学会論文誌 *D-II*, **J77**(8), pp.1663–1670, 1994.

[100] 筒口 拳, 末永 康仁, 渡部 保日児, 曽根原 登, “人物像歩行アニメーションにおける腕の動作生成方法”, 情報処理学会「グラフィクスと *CAD*」研究報告, **J96**(77), pp.55–60, 1996.

[101] 筒口 拳, 末永 康仁, 渡部 保日児, 下原 勝憲, “3 次元シーン内の人物像歩行動作生成システム”, 情報処理学会論文誌, **38**(4), pp.787–796, 1997.

[102] 筒口 拳, 青木 政勝, 曽根原 登, “遠隔協調作業環境による 3 次元コンピュータ・アニメーションの生成”, 情報処理学会第 *55* 回全国大会講演論文集 *(4)*, pp.270-271 (4AD-1), September 1997.

[103] 筒口 拳, 曽根原 登, “人流情報表示のための歩行動作の生成”, 電子情報通信学会秋季ソサイエティ大会（*ITS* シンポジウム）, pp.(SD-2-2-16), October 1998.

[104] 筒口 拳, 松田 達樹, 山田 正紀, 杉山 和弘, “WorldWideWalk^{TM}: 人物像歩行動作生成システム”, *NTT R & D*, **48**(12), pp.895–900, 1999.

[105] K. Tsutsuguchi, K. Sugiyama and N. Sonehara, “The Motion Generation of Pedestrians as Avatars and Crowds of People”, T. Ishida and K. Ishida (Eds.): *Digital Cities*, LNCS 1756, Springer-Verlag, pp.275–287, 2000. (Proceedings of Kyoto Meeting on Digital Cities 1999).

[106] 筒口 拳, 青木 政勝, 新谷 幹夫, “人物画像からの 3 次元歩行アニメーションの生成”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.291 (D-12-121), March 2000.

[107] K. Tsutsuguchi, S. Shimada, Y. Suenaga, N. Sonehara and S. Ohtsuka, “Human Walking Animation based on Foot Reaction Force in the Three-dimensional Virtual World”, *The Journal of Visualization and Computer Animation*, **11**(1), pp.3–16, 2000.

[108] M. Unuma, K. Anjyo and R. Takeuchi, “Fourier Principles for Emotion-based Human Figure Animation”, *Proceedings of SIGGRAPH 1995*, pp.91–96, 1995.

[109] 鵜沼 宗利, 安生 健一, 武内 良三, “群衆行動のモデリング – 仮想都市空間における人の群れと環境とのインタラクション –”, 電気学会論文誌 *C*, **115**(2), pp.212–221, 1995.

[110] M. Vukobratovic, 歩行ロボットと人工の足, 加藤一郎, 山下忠 訳, 日刊工業新聞社, 1975.

[111] M. Vukobratovic, ロボットの手 –力学と運動 –, 加藤一郎 監訳, 日刊工業新聞社, 1979.

[112] J. Wilhelms, “Virya- A Motion Control Techniques for Kimematic, Dynamic Animation”, *Proceedings of Graphics Interface 1986*, pp.141–146, 1986.

[113] J. Wilhelms, “Using Dynamic Analysis for Realistic Animation of Articulated Bodies”, *IEEE Computer Graphics and Aplications*, pp.12–27, June 1987.

[114] A. Witkin and Z. Popović, “Mortion Warping”, *Proceedings of SIGGRAPH 1995*, pp.105–108, 1995.

[115] M. Woo, J. Neider and T. Davis, *OpenGL Programming Guide Second Edition*, Addison-Wesley, 1997.

[116] D. Zeltzer, “Motor Control Techniques for Figure Animation”, *IEEE Computer Graphics and Applications*, **9**, pp.53—59, 1982.

[117] D. Zeltzer, “Motor Problem Solving for Three Dimentional Computer Animation”, *SIGGRAPH 1990 Course Notes*, pp.147—152, 1990.

付録 A

C/C++ インプリメンテーション

A.1 人体パーツのヘッダファイル

以下は本研究を C/C++ 言語で実装する際の，人物像構造パーツのマクロ定義を行うヘッダファイルである．

```
#ifndef _WWWALK_BODY_H_
#define _WWWALK_BODY_H_

#define BODY_BGN (0)
#define BODY_DLIST_BGN (BODY_BGN + 0 )
#define BODY_LENGTH_BGN (BODY_BGN + 0 )
#define BODY_ORIGIN_BGN (BODY_BGN + 0 )
#define BODY_NAME_BGN (BODY_BGN + 0 )
// 共通部分
#define BODY_PELVIS (BODY_NAME_BGN + 0)  // 骨盤
#define BODY_LW_BUST (BODY_NAME_BGN + 1)  // 腹部
#define BODY_UP_BUST (BODY_NAME_BGN + 2)  // 胸部
#define BODY_SHOULDER (BODY_NAME_BGN + 3)  // 肩部
#define BODY_NECK (BODY_NAME_BGN + 4)  // 首部
#define BODY_HEAD (BODY_NAME_BGN + 5)  // 頭部
// 左
#define BODY_THIGH_L (BODY_NAME_BGN + 6)  // 左大腿部
#define BODY_LEG_L (BODY_NAME_BGN + 7)  // 左下腿部
#define BODY_FOOT_L (BODY_NAME_BGN + 8)  // 左足部
#define BODY_TOE_L (BODY_NAME_BGN + 9)  // 左爪先部
#define BODY_UP_ARM_L (BODY_NAME_BGN + 10) // 左上腕部
#define BODY_LW_ARM_L (BODY_NAME_BGN + 11) // 左前腕部
#define BODY_HAND_L (BODY_NAME_BGN + 12) // 左手部
// 右
#define BODY_THIGH_R (BODY_NAME_BGN + 13) // 右腿部
#define BODY_LEG_R (BODY_NAME_BGN + 14) // 右下腿部
```

```
#define BODY_FOOT_R (BODY_NAME_BGN + 15) // 右足部
#define BODY_TOE_R (BODY_NAME_BGN + 16) // 右爪先部
#define BODY_UP_ARM_R (BODY_NAME_BGN + 17) // 右上腕部
#define BODY_LW_ARM_R (BODY_NAME_BGN + 18) // 右前腕部
#define BODY_HAND_R (BODY_NAME_BGN + 19) // 右手部
// end flag & alias
#define BODY_NAME_END (BODY_NAME_BGN + 20 )
#define BODY_DLIST_END (BODY_NAME_END + 0 )
// 末端部分
#define BODY_TOP_NAME_BGN (BODY_NAME_END + 0 )
#define BODY_HEAD_TOP (BODY_TOP_NAME_BGN + 0) // 頭頂部
#define BODY_TOE_L_TOP (BODY_TOP_NAME_BGN + 1) // 左爪先の先端部
#define BODY_HEEL_L_TOP (BODY_TOP_NAME_BGN + 2) // 左踵の先端部
#define BODY_HAND_L_TOP (BODY_TOP_NAME_BGN + 3) // 左手の先端部
#define BODY_TOE_R_TOP (BODY_TOP_NAME_BGN + 4) // 右爪先の先端部
#define BODY_HEEL_R_TOP (BODY_TOP_NAME_BGN + 5) // 右踵の先端部
#define BODY_HAND_R_TOP (BODY_TOP_NAME_BGN + 6) // 右手の先端部
#define BODY_TOP_NAME_END (BODY_TOP_NAME_BGN + 7)
#define BODY_ORIGIN_END (BODY_TOP_NAME_END + 0 )
// 以下は長さのみ
#define BODY_LEN_BGN        (BODY_NAME_END + 0 )//
#define BODY_H_FOOT_L       (BODY_LEN_BGN + 0)  // 足の回転中心の地面に対する高さ
#define BODY_H_TOE_L        (BODY_LEN_BGN + 1)  // 爪先の回転中心の地面に対する高さ
#define BODY_W_FOOT_L       (BODY_LEN_BGN + 2)  // 足と爪先の回転中心の水平距離
#define BODY_W_TOE_L        (BODY_LEN_BGN + 3)  // 爪先の水平長さ
#define BODY_H_FOOT_R       (BODY_LEN_BGN + 4)  // 足の回転中心の地面に対する高さ
#define BODY_H_TOE_R        (BODY_LEN_BGN + 5)  // 爪先の回転中心の地面に対する高さ
#define BODY_W_FOOT_R       (BODY_LEN_BGN + 6)  // 足と爪先の回転中心の水平距離
#define BODY_W_TOE_R        (BODY_LEN_BGN + 7)  // 爪先の水平長さ
#define BODY_FEET_DISTANCE (BODY_LEN_BGN + 8)  // 両足間の距離
#define BODY_H_PELVIS       (BODY_LEN_BGN +  9)  // 両脚付根の中点と人体中心の距離
#define BODY_W_PELVIS       (BODY_LEN_BGN + 10)  // 両足の付根の間の距離
#define BODY_H_SHOULDER     (BODY_LEN_BGN + 11)  // 両腕付根の中点と回転中心の距離
#define BODY_W_SHOULDER     (BODY_LEN_BGN + 12)  // 両腕の付根の間の距離
//
#define BODY_LEN_END        (BODY_LEN_BGN + 13)
#define BODY_LENGTH_END     (BODY_LEN_END + 0 )
#define BODY_END            (BODY_LENGTH_END + 0 )

// パーツ数 (== ディスプレイリストの数)
#define BODY_NAME_NUMBER ( BODY_NAME_END - BODY_NAME_BGN )
// ディスプレイリストの数
#define BODY_DLIST_NUMBER ( BODY_DLIST_END - BODY_DLIST_BGN )
// 長さの数
#define BODY_LENGTH_NUMBER ( BODY_LENGTH_END - BODY_LENGTH_BGN )
```



```
// 回転中心の数
#define BODY_ORIGIN_NUMBER ( BODY_ORIGIN_END - BODY_ORIGIN_BGN )

/* ------------------------------------------------------------------- *
   右手座標系で, X 軸は体から前方に向かう方向, Y 軸は鉛直上向き方向,
   Z 軸は人物像の左から右へ向かう方向

■ 人体ファイル ■   * は関節 (回転中心) を示す

                        番号: パーツの名称       意味
                        ------------------------------------
          [5]            // 中央
           *               0: BODY_PELVIS       腰部
          [4]              1: BODY_LW_BUST      腹部
           |               2: BODY_UP_BUST       胸部
     *-----[3]-----*       3: BODY_SHOULDER     肩部
     |     |     |         4: BODY_NECK         首部
    [17]   [2]   [10]      5: BODY_HEAD         頭部
     |     |     |        // 左側
     *     *     *         6: BODY_THIGH_L      大腿部・左
     |     |     |         7: BODY_LEG_L        下腿部・左
    [18]   |    [11]       8: BODY_FOOT_L       足部・左
     |    [1]    |         9: BODY_TOE_L        爪先部・左
  [19]*    |    *[12]     10: BODY_UP_ARM_L     上腕部・左
       *---[0]---*         11: BODY_LW_ARM_L     前腕部・左
       |         |         12: BODY_HAND_L       手部・左
     [13]       [6]       // 右側
       |         |         13: BODY_THIGH_R      大腿部・右
       *         *         14: BODY_LEG_R        下腿部・右
       |         |         15: BODY_FOOT_R       足部・右
     [14]       [7]        16: BODY_TOE_R        爪先部・右
       |         |         17: BODY_UP_ARM_R     上腕部・右
       *         *[8]*[9]  18: BODY_LW_ARM_R     前腕部・右
      [15]*[16]            19: BODY_HAND_R       手部・右
 * ------------------------------------------------------------------- */
#endif /* ! _WWWALK_BODY_H_ */
```

A.2 人体の位置および関節角度データのヘッダファイル

以下は本研究を C 言語（C++ 言語）で実装する際の，歩行データの型の定義を行うヘッダファイルである．

```
#ifndef _WWWALK_WALKING_DATA_
#define _WWWALK_WALKING_DATA_
// 構造体定義
typedef struct _coord3f { /* vector */
        float  x;
        float  y;
        float  z;
} Coord3d, coord3d, Coord4dv, coord4dv;

typedef struct _coord3d {        /* vector */
        double  x;
        double  y;
        double  z;
} Coord3d, coord3d, Coord4dv, coord4dv;

typedef struct _coord4d {        /* vector */
        double  x;
        double  y;
        double  z;
        double  a;        // 角度など
} Coord4d, coord4d, Coord4dv, coord4dv;

typedef struct _coord4f {        /* vector */
        float  x;
        float  y;
        float  z;
        float  a;
} Coord4f, coord4f, Coord4fv, coord4fv;

// Body Data (Height, Weight, Stride, Speed, Freq.)
typedef struct _body_data {
        double  height;    // 身長
        double  weight;    // 体重
        double  stride;    // デフォルトの歩幅
        double  speed;     // デフォルトの水平歩行速度
        double  freq;      // デフォルトの歩数／分
} BodyData;

// 人体の姿勢を表すデータ
//
// 各軸回りの関節の回転角度；回転の自由度が 3 でない関節の，余分な角度は無視する
typedef struct _s_MIDANGLE {        // 共通部分
```



```
        coord3f  pelvis;        // 骨盤の回転
        coord3f  lwbust;        // 腹部の回転
        coord3f  upbust;        // 胸部の回転
        coord3f  shoulder;      // 肩部の回転
        coord3f  neck;          // 首部の回転
        coord3f  head;          // 頭部の回転
} MIDANGLE;
typedef struct _s_SIDEANGLE {       // 左または右
        coord3f  hip;           // 股関節の回転角度
        coord3f  knee;          // 膝関節の回転角度
        coord3f  leg;           // 下腿部が鉛直線に対してなす角度
        coord3f  ankle;         // 足首の関節角度；下腿部と足部がなす角度
        coord3f  toe;           // 爪先部の，足部に対する角度
        coord3f  arm;           // 上腕部の肩に対する回転角度
        coord3f  elbow;         // 肘部の回転角度
        coord3f  wrist;         // 手首部の回転角度
} SIDEANGLE;

// 1 フレームあたりの歩行データ
typedef struct _s_FRAMEDATA {
        float     x;            // 人体原点の X 座標値
        float     y;            // 人体原点の Y 座標値
        float     z;            // 人体原点の Z 座標値
        MIDANGLE  M;            // 共通部分の関節角度
        SIDEANGLE L;            // 左側の関節角度
        SIDEANGLE R;            // 右側の関節角度
} FRAMEDATA;

// 1 歩あたりの歩行データ
typedef struct _s_STEPDATA {
        int         frames;     // フレーム数
        int         stand;      // 支持脚; 0: 左支持, 1: 右支持
        int         frames;     // 全フレーム数（持続時間を表す）
        float       stride;     // 歩幅
        FRAMEDATA * fdata;      // 実データへのポインタ
} STEPDATA;

// 歩行開始から終了までの歩行データ
typedef struct _s_WALKDATA {
        int         steps;      // 歩数
        int         frames;     // 全フレーム数
        STEPDATA * sdata;       // 実データへのポインタ
} WALKDATA;
#endif        // !_WWWALK_WALKING_DATA_
```

A.3 腰部を基準にした人物像の描画を行うコード

以下は本研究を C 言語（C++ 言語）で実装する際の，腰部に原点を置く場合の 1 フレームあたりの歩行動作を描画する疑似コードである．ここで，PushMatrix/PopMatrix は座標変換のスタックのプッシュ／ポップを表し，Translate は平行移動を表し，Rotate は座標軸まわりの回転を表す．また，CallList は，ディスプレイリストに記憶された部位を描画することを表す．

```
/* ----- カレントフレームの表示 ----- */
PushMatrix();
Translate( その一歩の原点への移動 );
Rotate( 人体の向きに回転 );

PushMatrix(); { // 人体構造の記述
   Translate( 人体の原点へ );
   Translate( 腰部中心へ );
   PushMatrix(); {
      Rotate( 腰部の回転 );
      CallList( 腰部 );
      // 左脚部
      PushMatrix(); {     // 左大腿部の描画
         Translated( 左股関節へ );
         Rotate( 股関節の回転 );
         CallList( 左大腿部 );
         PushMatrix(); {     // 左下腿部の画描
            Translate( 左膝関節へ );
            Rotate( 左膝関節の回転 );
            CallList( 左下腿部 );
            PushMatrix(); {     // 左足部の描画
               Translate( 左足首部へ );
               Rotate( 左足首関節の回転 );
               CallList( 左足部 );
               PushMatrix(); {     // 左爪先部の描画
                  Translate( 左爪先関節へ );
                  Rotate( 左爪先関節の回転 );
                  CallList( 左爪先部 );
               } PopMatrix();
            } PopMatrix();
         } PopMatrix();
      } PopMatrix();
   } PopMatrix();     // 原点は，回転した腰部の中心にある
} PopMatrix();     // 原点は，進行方向を X 軸とする腰部の中心にある
// 右脚部
PushMatrix(); {     // 右大腿部の描画
   Translated( 右股関節へ );
   Rotate( 股関節の回転 );
   CallList( 右大腿部 );
   PushMatrix(); {     // 右下腿部の画描
```



```
            Translate( 右膝関節へ );
            Rotate( 右膝関節の回転 );
            CallList( 右下腿部 );
            PushMatrix(); {     // 右足部の描画
               Translate( 右足首部へ );
               Rotate( 右足首関節の回転 );
               CallList( 右足部 );
               PushMatrix(); {     // 右爪先部の描画
                  Translate( 右爪先関節へ );
                  Rotate( 右爪先関節の回転 );
                  CallList( 右爪先部 );
               } PopMatrix();
            } PopMatrix();
         } PopMatrix();
      } PopMatrix();     // 原点は，回転した腰部の中心にある
   } PopMatrix();     // 原点は，進行方向を X 軸とする腰部の中心にある
   // 上半身の描画
   PushMatrix(); {
      // 腹部の描画
      Translate( 腹部の原点へ );
      Rotate( 腹部関節の回転 );
      CallList( 腹部 );
      PushMatrix(); {
        // 胸部の描画
         Translate( 胸部の原点へ );
         Rotate( 胸部関節の回転 );
         CallList( 胸部 );
         PushMatrix(); {
            Translate( 肩部の原点へ );
            // この段階で，原点は回転前の肩部にある
            PushMatrix(); {
               // 首部の描画
               Translate( 首関節へ );
               Rotate( 首関節の回転 );
               CallList( 首部 );
               PushMatrix(); {
                  // 頭部の描画
                  Translate( 頭部の原点へ );
                  Rotate( 頭部の回転 );
                  CallList( 頭部 );
               } PopMatrix();
            } PopMatrix();
            // この段階で，原点は開店前の肩部に戻る
            Rotate( 肩の回転 );
            CallList( 肩部 );
```

```
//
// 腕部の描画
//
PushMatrix(); {     // 左上腕部の描画
   Translate( 左肩関節へ );
   Rotate( 左上腕部の回転 );
   CallList( 左上腕部 );
   PushMatrix(); {     // 左前腕部の描画
      Translate( 左肘関節へ );
      Rotate( 左肘関節の回転 );
      CallList( 左前腕部 );
      PushMatrix(); {     // 左手部の描画
         Translate( 左手首関節へ );
         Rotate( 左手首関節の回転 );
         CallList( 左手部 );
      } PopMatrix();
   } PopMatrix();
} PopMatrix();
// 原点は回転後の肩中心にある
PushMatrix(); {     // 右上腕部の描画
   Translate( 右肩関節へ );
   Rotate( 右上腕部の回転 );
   CallList( 右上腕部 );
   PushMatrix(); {     // 右前腕部の描画
      Translate( 右肘関節へ );
      Rotate( 右肘関節の回転 );
      CallList( 右前腕部 );
      PushMatrix(); {     // 右手部の描画
         Translate( 右手首関節へ );
         Rotate( 右手首関節の回転 );
         CallList( 右手部 );
      } PopMatrix();
   } PopMatrix();
} PopMatrix();     // 原点は回転後の肩の中心
} PopMatrix();     // 原点は回転前の肩の中心
} PopMatrix();     // 胸部中心
} PopMatrix();     // 腰部中心
} PopMatrix();     // 地面上
PopMatrix();     // ワールド座標原点へ
```

付録 B

Java/Java3D インプリメンテーション

B.1 人体パーツを定義するクラス

以下は本研究を Java 言語で実装する際の，人物像構造パーツのクラス定義を行うソースファイルである．

```
// Name.java
public class Name {
    public static final int NAME_BGN = 0;
    public static final int LENGTH_BGN = NAME_BGN + 0;
    public static final int ORIGIN_BGN = NAME_BGN + 0;
    /* --- Name(ディスプレイリスト)はここから --- */
    // 中央部
    public static final int PELVIS   = NAME_BGN + 0; // 腰部
    public static final int LW_BUST  = NAME_BGN + 1; // 腹部
    public static final int UP_BUST  = NAME_BGN + 2; // 胸部
    public static final int SHOULDER = NAME_BGN + 3; // 肩部
    public static final int NECK     = NAME_BGN + 4; // 首部
    public static final int HEAD     = NAME_BGN + 5; // 頭部
    // 左
    public static final int THIGH_L  = NAME_BGN + 6; // 左大腿部
    public static final int LEG_L    = NAME_BGN + 7; // 左下腿部
    public static final int FOOT_L   = NAME_BGN + 8; // 左足部
    public static final int TOE_L    = NAME_BGN + 9; // 左爪先部
    public static final int UP_ARM_L = NAME_BGN + 10; // 左上腕部
    public static final int LW_ARM_L = NAME_BGN + 11; // 左前腕部
    public static final int HAND_L   = NAME_BGN + 12; // 左手部
    // 右
    public static final int THIGH_R  = NAME_BGN + 13; // 右大腿部
    public static final int LEG_R    = NAME_BGN + 14; // 右下腿部
    public static final int FOOT_R   = NAME_BGN + 15; // 右足部
    public static final int TOE_R    = NAME_BGN + 16; // 右左爪先部
    public static final int UP_ARM_R = NAME_BGN + 17; // 右上腕部
    public static final int LW_ARM_R = NAME_BGN + 18; // 右前腕部
```

```
    public static final int HAND_R   = NAME_BGN + 19; // 右手部
    public static final int NAME_END = NAME_BGN + 20 ; // alias
    /* --- 回転中心 (ローカル座標) はまだ続く --- */
    public static final int TOP_NAME_BGN = NAME_END + 0 ;
    public static final int HEAD_TOP     = TOP_NAME_BGN + 0; // 頭頂部
    public static final int TOE_L_TOP    = TOP_NAME_BGN + 1; // 左爪先端
    public static final int HEEL_L_TOP   = TOP_NAME_BGN + 2; // 左踵先端
    public static final int HAND_L_TOP   = TOP_NAME_BGN + 3; // 左手先端
    public static final int TOE_R_TOP    = TOP_NAME_BGN + 4; // 右爪先端
    public static final int HEEL_R_TOP   = TOP_NAME_BGN + 5; // 右踵先端
    public static final int HAND_R_TOP   = TOP_NAME_BGN + 6; // 右手先端
    public static final int TOP_NAME_END = TOP_NAME_BGN + 7; // alias
    public static final int ORIGIN_END   = TOP_NAME_END + 0 ;
    /* --- 以下は長さのみ．C() は回転中心を示す．C(足) は足首である．*/
    public static final int LEN_BGN  = NAME_END + 0; // alias
    public static final int H_FOOT_L = LEN_BGN + 0;  // C(足) の地面からの高さ
    public static final int H_TOE_L  = LEN_BGN + 1;  // C(爪先) の地面からの高さ
    public static final int W_FOOT_L = LEN_BGN + 2;  // C(足) と C(爪先) の水平距離
    public static final int W_TOE_L  = LEN_BGN + 3;  // 爪先の水平長さ
    public static final int H_FOOT_R = LEN_BGN + 4;  // C(足) の地面からの高さ
    public static final int H_TOE_R  = LEN_BGN + 5;  // C(爪先) の地面からの高さ
    public static final int W_FOOT_R = LEN_BGN + 6;  // C(足) と C(爪先) の水平距離
    public static final int W_TOE_R  = LEN_BGN + 7;  // 爪先の水平長さ
    public static final int FEET_DISTANCE = LEN_BGN+8; // 両足間の距離
    // 両脚付根の中点と人体中心の距離
    public static final int H_PELVIS   = LEN_BGN +  9;
    // 両足の付根の間の距離
    public static final int W_PELVIS   = LEN_BGN + 10;
    // 両腕付根の中点と回転中心の距離
    public static final int H_SHOULDER = LEN_BGN + 11;
    // 両腕の付根の間の距離
    public static final int W_SHOULDER = LEN_BGN + 12;
    public static final int LEN_END    = LEN_BGN + 13;
    public static final int LENGTH_END = LEN_END + 0 ;
    // パーツ数
    public static final int NAME_NUMBER   = NAME_END - NAME_BGN;
    // 長さの数
    public static final int LENGTH_NUMBER = LENGTH_END - LENGTH_BGN;
    // 回転中心の数
    public static final int ORIGIN_NUMBER = ORIGIN_END - ORIGIN_BGN;
    // コンストラクタ; 何もしない
    public Name() {;}
}
```


B.2 人体の位置および関節角度データを定義するクラス

以下は本研究を Java 言語で実装する際の，歩行データのクラス定義を行うソースファイルである．このクラスでは，Java3D を用いている．

B.2.1 フレーム毎のデータ

```
// FrameData.java
package walkdata;
public class FrameData {
    /** --- Class in Class --- */
    public class MidAngle {
        public Vector3d pelvis = new Vector3d(0.0, 0.0, 0.0);    // 腰部
        public Vector3d lwbust = new Vector3d(0.0, 0.0, 0.0);    // 腹部
        public Vector3d upbust = new Vector3d(0.0, 0.0, 0.0);    // 胸部
        public Vector3d shoulder = new Vector3d(0.0, 0.0, 0.0); // 肩部
        public Vector3d neck = new Vector3d(0.0, 0.0, 0.0);      // 首部
        public Vector3d head = new Vector3d(0.0, 0.0, 0.0);      // 頭部
        // Constructor; Nothing to do.
        public MidAngle() {;}
    }
    public class SideAngle {
        public Vector3d hip   = new Vector3d(0.0, 0.0, 0.0);     // 股関節
        public Vector3d knee  = new Vector3d(0.0, 0.0, 0.0);     // 膝関節
        public Vector3d ankle = new Vector3d(0.0, 0.0,70.0d);    // 足首関節
        public Vector3d toe   = new Vector3d(0.0, 0.0, 0.0);     // 足と爪先の間
        public Vector3d arm   = new Vector3d(0.0, 0.0, 0.0);     // 上腕部
        public Vector3d elbow = new Vector3d(0.0, 0.0, 0.0);     // 肘関節
        public Vector3d wrist = new Vector3d(0.0, 0.0, 0.0);     // 手首関節
        // Constructor; Nothing to do.
        public SideAngle() {;}
    }
    /** --- Public Fields --- */
    public Vector3d center = new Vector3d(0.0, 0.0, 0.0);       // 人体中心位置
    public MidAngle  M = new MidAngle();         // 中央部の角度
    public SideAngle L = new SideAngle();        // 左半身の角度
    public SideAngle R = new SideAngle();        // 右半身の角度
    // Constructor; Nothing to do.
    public FrameData() {;}
}
```

B.2.2 1歩毎のデータ

```
// StepData.java
package walkdata;
import walkdata.FrameData;
```

```
public class StepData {
    /** --- Public & Static --- */
    public static final int DEFAULT_FRAME_NUMBER = 36;
    public static final int LEFT_STAND  = 1;
    public static final int RIGHT_STAND        = 2;
    // 状態
    public static final int LEFT_STAND_START  =  0;
    public static final int LEFT_STAND_WALK   =  1;
    public static final int LEFT_STAND_STOP   =  2;
    public static final int RIGHT_STAND_STOP  =  3;
    public static final int RIGHT_STAND_WALK  =  4;
    public static final int RIGHT_STAND_START =  5;
    public static final int NO_STATUS         = 10;
    public static final int LEFT_STAND_LTURN  = 11;
    public static final int LEFT_STAND_RTURN  = 12;
    public static final int RIGHT_STAND_LTURN = 13;
    public static final int RIGHT_STAND_RTURN = 14;
    public static final int STRAIGHT_FORWARD  = 15;
    /** --- Public Data Fields --- */
    public int   state   = LEFT_STAND_START;        // 開始／停止状態
    public int   turn    = STRAIGHT_FORWARD;        // 回転状態
    public int   stand   = LEFT_STAND;              // 支持脚
    public int   frames  = DEFAULT_FRAME_NUMBER;    // 全フレーム数
    public double stride = 0.80;                    // 歩幅
    public FrameData[] frame;                       // 実データへの参照
    /** --- Constructor --- */
    public StepData() {
        this.frame  = new FrameData[frames];
        for ( int i=0; i<this.frames; i++ ) {
            this.frame[i] = new FrameData();
        }
    }
    public StepData( int frame_number ) {
        this.frames = frame_number;
        this.frame  = new FrameData[frame_number];
        for ( int i=0; i<frame_number; i++ ) {
            this.frame[i] = new FrameData();
        }
    }
    /** --- Output methods --- */
    public int getState() { return this.state; }
    public int getTurn () { return this.turn; }
    public int getStand() { return this.stand; }
    public int getFrames() { return this.frames; }
    public double getStride() { return this.stride; }
}
```

B.2.3 歩行開始から歩行終了までのデータ

```
// WalkData.java
package walkdata;
import walkdata.FrameData;
```



```
import walkdata.StepData;

public class WalkData {
    /** --- Public Fields --- */
    public int steps = 2;        // start & stop
    public StepData[] step;
    /** --- Constructor --- */
    public WalkData() {
        for (int i=0; i<this.steps; i++ ) {
            this.step[i] = new StepData();
        }
    }
    public WalkData( int n ) {
        this.steps = n;
        for (int i=0; i<this.steps; i++ ) {
            this.step[i] = new StepData();
        }
    }
    /** --- Output Methods --- */
    public int getSteps() { return this.steps; }
}
```

B.3 腰部を基準にした人物像の描画を行う疑似コード

以下は本研究を Java 言語で実装する際の，1フレーム単位で歩行動作を表示するクラスにおける，描画メソッドの疑似コードである．このクラスでは，Java3D を用いている．

```
// 人体のパーツを描画する TransformGroup の派生クラス
public Part[] part = new Part[Name.NAME_NUMBER];

// 人体のシーングラフを格納する
public TransformGroup center = new TransformGroup();

// 人体の姿勢を描画するメソッド
public TransformGroup pose( FrameData f ) {

    Transform3D rot = new Transform3D();          // 全体の平行移動
    Transform3D all = new Transform3D();          // 全体の回転
    all.setTranslation(人物像の原点);             // 人物像原点に平行移動
    rot.rotY(人物像の向き);                       // 人物像の方向に回転
    all.mul(rot, all);
    this.center.setTransform(all);

    // 腰部; 重心に原点を移動
    part[Name.PELVIS].translate( 腰部の原点へ );
    part[Name.PELVIS].rotate( 腰部の回転 );
    part[Name.PELVIS].bindNoShapeTo( this.center );

    // 腹部
    part[Name.LW_BUST].translate( 腹部の原点へ );
    part[Name.LW_BUST].rotate( 腹部の回転 );
    part[Name.LW_BUST].bindTo((TransformGroup)this.part[Name.PELVIS]);
    // 胸部
    part[Name.UP_BUST].translate( 胸部の原点へ );
    part[Name.UP_BUST].rotate( 胸部の回転 );
    part[Name.UP_BUST].bindTo((TransformGroup)this.part[Name.LW_BUST]);
    // 肩部
    part[Name.SHOULDER].translate( 肩部の原点へ );
    part[Name.SHOULDER].rotate( 肩部の回転 );
    part[Name.SHOULDER].bindNoShapeTo
          ((TransformGroup)this.part[Name.UP_BUST]);
    // 首部
    part[Name.NECK].translate( 首部の原点へ );
    part[Name.NECK].rotate( 首部の回転 );
    part[Name.NECK].bindTo((TransformGroup)this.part[Name.SHOULDER]);
    // 頭部
```



```
    part[Name.HEAD].translate( 頭部の原点へ );
    part[Name.HEAD].rotate( 頭部の回転 );
    part[Name.HEAD].bindTo((TransformGroup)this.part[Name.NECK]);

    // 左大腿部
    part[Name.THIGH_L].translate( 左大腿部の原点へ );
    part[Name.THIGH_L].rotate( 左股関節の回転 );
    part[Name.THIGH_L].bindTo((TransformGroup)this.part[Name.PELVIS]);
    // 左下腿部
    part[Name.LEG_L].translate( 左下腿部の原点へ );
    part[Name.LEG_L].rotate( 左膝関節の回転 );
    part[Name.LEG_L].bindTo((TransformGroup)this.part[Name.THIGH_L]);
    // 左足部
    part[Name.FOOT_L].translate( 左足部の原点へ );
    part[Name.FOOT_L].rotate( 左足首関節の回転 );
    part[Name.FOOT_L].bindTo((TransformGroup)this.part[Name.LEG_L]);
    // 左爪先部
    part[Name.TOE_L].translate( 左爪先部の原点へ );
    part[Name.TOE_L].rotate( 左爪先関節の回転 );
    part[Name.TOE_L].bindTo((TransformGroup)this.part[Name.FOOT_L]);
    // 左上腕部
    part[Name.UP_ARM_L].translate( 左上腕部の原点へ );
    part[Name.UP_ARM_L].rotate( 左上腕部関節の回転 );
    part[Name.UP_ARM_L].bindTo((TransformGroup)this.part[Name.SHOULDER]);
    // 左下腕部
    part[Name.LW_ARM_L].translate( 左前腕部の原点へ );
    part[Name.LW_ARM_L].rotate( 左肘関節の回転 );
    part[Name.LW_ARM_L].bindTo((TransformGroup)this.part[Name.UP_ARM_L]);
    // 左手部
    part[Name.HAND_L].translate( 左手部の原点へ );
    part[Name.HAND_L].rotate( 左手首関節の回転 );
    part[Name.HAND_L].bindTo((TransformGroup)this.part[Name.LW_ARM_L]);

    // 右大腿部
    part[Name.THIGH_R].translate( 右大腿部の原点へ );
    part[Name.THIGH_R].rotate( 右股関節の回転 );
    part[Name.THIGH_R].bindTo((TransformGroup)this.part[Name.PELVIS]);
    // 右下腿部
    part[Name.LEG_R].translate( 右下腿部の原点へ );
    part[Name.LEG_R].rotate( 右膝関節の回転 );
    part[Name.LEG_R].bindTo((TransformGroup)this.part[Name.THIGH_R]);
    // 右足部
    part[Name.FOOT_R].translate( 右足部の原点へ );
    part[Name.FOOT_R].rotate( 右足首関節の回転 );
    part[Name.FOOT_R].bindTo((TransformGroup)this.part[Name.LEG_R]);
```

```
    // 右爪先部
    part[Name.TOE_R].translate( 右爪先部の原点へ );
    part[Name.TOE_R].rotate( 右爪先関節の回転 );
    part[Name.TOE_R].bindTo((TransformGroup)this.part[Name.FOOT_R]);
    // 右上腕部
    part[Name.UP_ARM_R].translate( 右上腕部の原点へ );
    part[Name.UP_ARM_R].rotate( 右上腕部関節の回転 );
    part[Name.UP_ARM_R].bindTo((TransformGroup)this.part[Name.SHOULDER]);
    // 右下腕部
    part[Name.LW_ARM_R].translate( 右前腕部の原点へ );
    part[Name.LW_ARM_R].rotate( 右肘関節の回転 );
    part[Name.LW_ARM_R].bindTo((TransformGroup)this.part[Name.UP_ARM_R]);
    // 右手部
    part[Name.HAND_R].translate( 右手部の原点へ );
    part[Name.HAND_R].rotate( 右手首関節の回転);
    part[Name.HAND_R].bindTo((TransformGroup)this.part[Name.LW_ARM_R]);

    return center;
}
```

本研究に関する業績

学術論文

筒口 拳, 境野 英朋, 渡部 保日児, “地形適応型歩行動作生成手法を用いた人物の歩行アニメーション”, 電子情報通信学会論文誌 *D-II*, **J77**(8), pp.1663–1670, 1994.

筒口 拳, 末永 康仁, 渡部 保日児, 下原 勝憲, “3 次元シーン内の人物像歩行動作生成システム”, 情報処理学会論文誌, **38**(4), pp.787–796, 1997.

K. Tsutsuguchi, S. Shimada, Y. Suenaga, N. Sonehara and S. Ohtsuka, “Human Walking Animation based on Foot Reaction Force in the Three-dimensional Virtual World”, *The Journal of Visualization and Computer Animation*, **11**(1), pp.3–16, 2000.

解説論文

筒口 拳, 松田 達樹, 山田 正紀, 杉山 和弘, “WorldWideWalk^{TM}: 人物像歩行動作生成システム”, *NTT R & D*, **48**(12), pp.895–900, 1999.

国際会議

K. Tsutsuguchi, H. Sakaino and Y. Watanabe, “Terrain Adaptive Human Walking”, *Insight Through Computer Graphics (Proceedings of CG International 1994)*, World Scientific, pp.180–191, 1996.

K. Tsutsuguchi, K. Sugiyama and N. Sonehara, “The Motion Generation of Pedestrians as Avatars and Crowds of People”, T. Ishida and K. Ishida (Eds.): *Digital Cities*, LNCS 1756, Springer-Verlag, pp.275–287, 2000.

全国大会・研究会

筒口 拳, “環境適応型歩行動作生成手法の検討”, 情報処理学会第 *46* 回全国大会講演論文集 *Number 2*, pp.467–468, 1993.

筒口 拳, 渡部 保日児, “環境適応型歩行動作生成手法”, 情報処理学会第 *47* 回全国大会講演論文集 *Number 2*, pp.343–344, 1993.

筒口 拳 and 渡部 保日児, “人物の歩行動作生成手法”, 「グラフィクスと *CAD*」シンポジウム講演論文集, pp.189 197, 1993.

筒口 拳, 末永 康仁, 渡部 保日児, “室内における人物像の歩行アニメーション”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.419 (D631), March 1996.

筒口 拳, 末永 康仁, 渡部 保日児, 曽根原 登, “人物像歩行アニメーションにおける腕の動作生成方法”, 情報処理学会「グラフィクスと *CAD*」研究報告, **J96**(77), pp.55–60, 1996.

筒口 拳, 末永 康仁, 渡部 保日児, 曽根原 登, マルチメディア実験における非圧縮画像転送方式の検討, 第 *152* 回画像電子学会研究会, pp.21–24 (960206), September 1996.

筒口 拳, 末永 康仁, 渡部 保日児, 曽根原 登, “人物像歩行アニメーション”, 第 *12* 回 *NICOGRAPH* 論文コンテスト論文集, pp.33–42, 1996.

筒口 拳, 花里 高宏, 末永 康仁, 曽根原 登, “動力学モデルに基づく歩行に付随する腕の運動生成方法”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.392 (D-12-185), March 1997.

筒口 拳, 青木 政勝, 曽根原 登, “遠隔協調作業環境による 3 次元コンピュータ・アニメーションの生成”, 情報処理学会第 *55* 回全国大会講演論文集 *(4)*, pp.270-271 (4AD-1), September 1997.

筒口 拳, 青木 政勝, 曽根原 登, “仮想環境における複数の人物像の歩行動作生成手法”, 電子情報通信学会春季総合大会（情報・システム *2* シンポジウム）, pp.481-482 (SD-2-2-16), March 1998.

筒口 拳, 曽根原 登, “人流情報表示のための歩行動作の生成”, 電子情報通信学会秋季ソサイエティ大会（*ITS* シンポジウム）, pp.(SD-2-2-16), October 1998.

筒口 拳, 青木 政勝, 新谷 幹夫, “人物画像からの 3 次元歩行アニメーションの生成”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.291 (D-12-121), March 2000.

解説記事等

“SGI Frequently Asked Questions”, ネットニュースに数回に渡り投稿,
http://faqs.jmas.co.jp/FAQs/sgi-faq-j/ など.

“OpenGL Frequently Asked Questions （日本語版）”,
http://www.nk-exa.co.jp/ andoh/opengl/faq/openglfaq.html など.

“The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3, 日本語翻訳版”,
http://www.nk-exa.co.jp/ andoh/opengl/glut/ など.

共著

青木 政勝, 筒口 拳, 曽根原 登, “編集履歴を用いたネットワーク上でのコンテンツ編集方法”, 電子情報通信学会秋季ソサイエティ大会（情報・システム *2*）, pp.286 (D-12-76), September 1997.

青木 政勝, 筒口 拳, 曽根原 登, “2 次元インタフェースによる 3 次元 CG 生成方法”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.308 (D-12-109), March 1998.

青木 政勝, 新谷 幹夫, 筒口 拳, “動力学を利用した静止画の動画像化に関する検討”, 電子情報通信学会秋季ソサエティ大会（情報・システム）, pp.289 (D-12-67), September 1998.

青木 政勝, 新谷 幹夫, 筒口 拳, 曽根原 登, “静止画の動画像化における動力学パラメータの指定”, 電子情報通信学会春季総合大会（情報・システム）, pp.321 (D-12-148), March 1999.

花里 高宏, 筒口 拳, 古川 貴雄, 曽根原 登, 清水 義雄, “Bezier 曲面を用いた衣服形状生成”, 電子情報通信学会春季総合大会（情報・システム *2*）, pp.386(D-12-179), March 1997.

花里 高宏, 筒口 拳, 古川 貴雄, 曽根原 登, 清水 義雄, “双 3 次 Bezier パッチの適応的分割を用いた衣服形状生成”, 情報処理学会「グラフィクスと *CAD*」研究報告, **97**(86), pp.61–66, August 1997.

佐伯 貴利, 古川 貴雄, 清水 義雄, 筒口 拳, 曽根原 登, “バンプマッピングを応用した柔軟な衣服表現法”, 情報処理学会「グラフィクスと *CAD*」研究報告, **98**(16), pp.17–22, Febrary 1998.

M. Shinya, M. Aoki, K. Tsutsuguchi and N. Konani, “Dynamic Texture: Physically Based 2D Animation”, *Proceedings of SIGGRAPH 1999 Conference Abstracts and Applications*, pp.239, August 1999.

山田 正紀, 筒口 拳, 新谷 幹夫, 曽根原 登, “Multiple-Baseline Stereo 法を用いた 3 次元映像獲得環境に関する実験”, 電子情報通信学会春季総合大会（情報・システム）, pp.(D-12-168), March 1999.